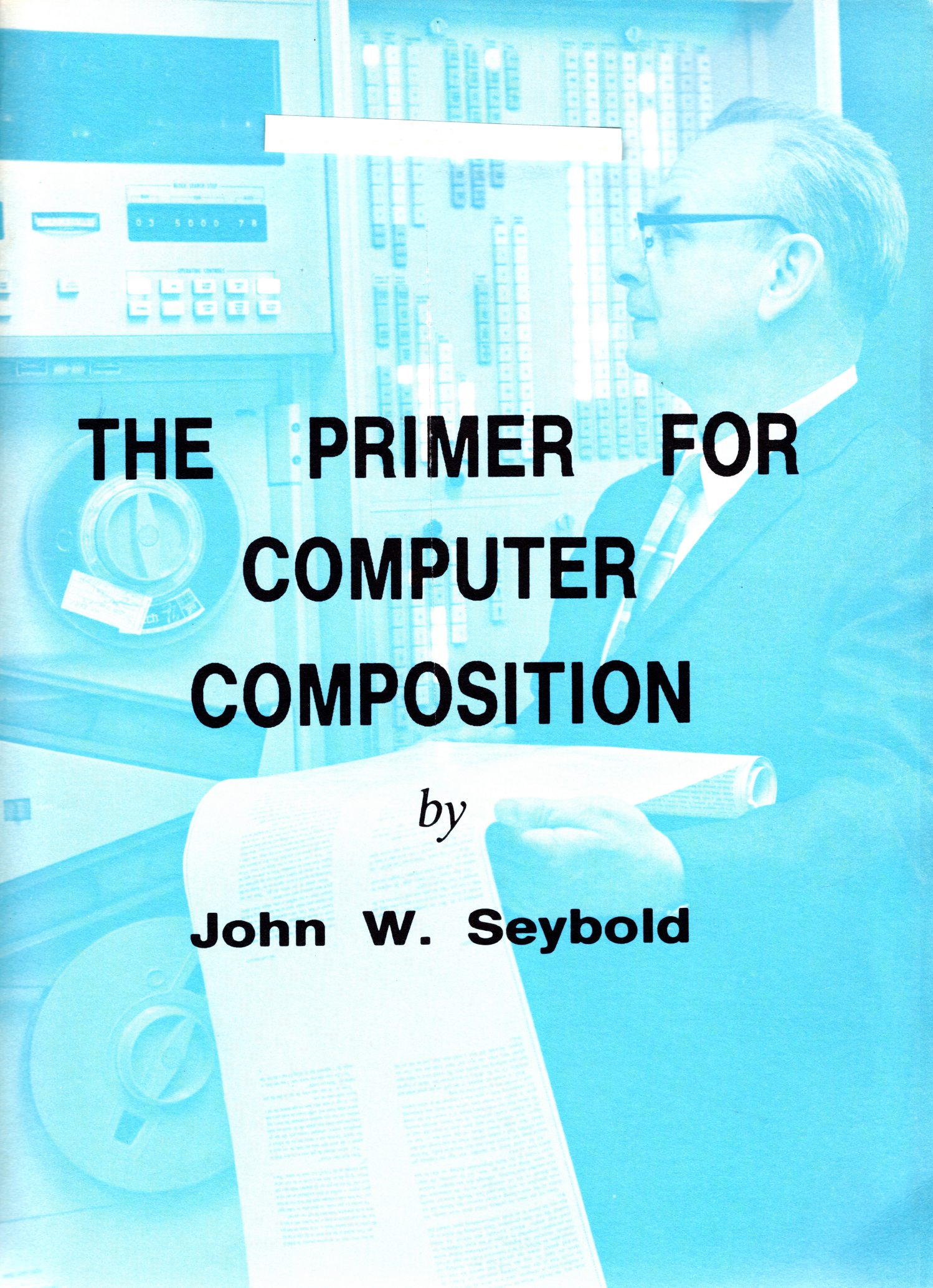


# THE PRIMER FOR COMPUTER COMPOSITION

*by*

**John W. Seybold**

With particular appreciation for the gentle patience of Linda in guiding this manuscript through the intricate channels of computer composition, and with gratitude to Harry, Ed, and the proofreaders at Rocappi for their help and advice.

*Copyright © 1971 by the Graphic Communications Computer Association, Printing Industries of America, Inc. All rights reserved.*

# THE PRIMER
# FOR
# COMPUTER COMPOSITION

**John W. Seybold**
*Consultant, John W. Seybold & Associates*

Graphic Communications Computer Association
Printing Industries of America, Inc.
Arlington, Va.

# Preface

Ever since the appearance in 1969 of John W. Seybold's book, *The Market for Computerized Composition,* which was published by the Graphic Communications Computer Association (then the Computer Section of PIA), the rapid expansion of the use of and demand for this production technology has increased significantly and with it the need for knowledge. Within the last year we have seen the discontinuance of production in the United States of hot metal typesetting machines, and a continued growth in the use of offset printing equipment. These major changes are writing a new chapter in the long history of printing, and when they are coupled with the ever-growing use of the computer as a tool for information creation and storage they establish a clear need for all persons who are versed in older composition technologies—from salesman to typographer—to understand the new role of the computer in their lives.

Moreover, the host of specialists in electronics who are being called upon to contribute their knowledge for the benefit of the graphic arts industry also need a source for reference in order to gain a full appreciation of the printer's art.

Hence, *The Primer for Computer Composition.* And John Seybold's unmatched knowledge of the field, his unique ability to conceptualize, and his profound understanding and empathy for people, make him particularly qualified to meet the task at hand.

We in graphic communications are indeed most fortunate to have this scholar in our midst.

Norman W. Scharpf,
*Executive Director,*
*GCCA/PIA*

# The Primer for Computer Composition

## Introduction

The word "primer" should be self-explanatory. My intention is to be as elementary as possible, and to assume no prior knowledge of the subject on the part of the reader.

On the other hand, it is not my intention to be superficial, but rather my hope is to deal in as lucid a manner as possible with the fundamental concepts necessary to an understanding of the unique properties of a typesetting system in which the computer plays a significant role.

The reader may wish to skip over those sections of this book which seem too elementary to him. On the other hand, he may gain new insights by following the presentation of the subject as it moves along.

This presentation is directed toward a readership made up of buyers of printing and composition, authors, editors, data processing experts and programmers, keyboard operators, mark-up people and all those whose living may be derived from or influenced by the new composition technology. Salesmen are by no means excluded, for theirs is the task of discovery and interpretation, and they will need some grasp of basic concepts.

I hope the reader will not be offended by the questions which are intended to serve as a convenient study guide. At least they will highlight what I think is important.

J.W.S.

CHAPTER I

# How Type Composition Differs From Typewriter Composition

## 1. Impact and Cold Composition

Typewriters produce a form of "cold composition" as distinguished from "hot type."[1] There are other forms of cold composition which produce character images by photographic or electronic means. A typewriter produces its image by the impact of a raised letter striking a ribbon and leaving a carbon or ink impression on the paper. Thus a typewriter produces "impact" composition.

The term "composition" would be used if the product of the typewriter were to be reproduced, as in a book or report, rather than to serve as "copy" (or "manuscript") for resetting. Composition implies some effort to arrange the material not only legibly but aesthetically in pages.

## 2. Character Widths (Escapement)

The type characters on most typewriters are all of the same width. The term "uniform widths" is used to describe the appearance of images so designed that each takes up the same amount of space, whether it is a punctuation mark, a lower case character or a capital (upper case) letter. When a typewriter key is struck the carriage, which includes the platen or roller and paper transport mechanism, moves from right to left by the space of one character. This is the "escapement" of the carriage and it is controlled by a gear and pinion device. On IBM Selectric typewriters it is the "golf ball" element which escapes and the platen remains fixed. In both cases it is possible to backspace one keystroke and then to replace one character (any character) with another.

However, there are some typewriters which produce a typescript consisting of letters and symbols not all of the same width. In this case the escapement mechanism is more complicated. The machine has to "know" what particular character is called for and what carriage escapement is associated with that character. The range of widths varies from a wide character of five units to a narrow one of two units. In backspacing to achieve a strikeover it is necessary to know the width of the character which was previously typed, for each stroke of the backspace key will return the carriage by only one unit.

As in the case of "regular" type—that is, hot metal type or type designed to look like hot metal—type characters are "proportional." This word is used to indicate that not all characters are the same width, but of varying widths which are designed to achieve a better—i.e., more readable—appearance, as well as to save space.

[1] Type is "hot" when it is cast from molten metal in typesetting.

Because of the limitations of typewriter mechanics and manufacturing costs, however, the degree of proportionality—i.e., the range of variations in widths of different characters—is limited, and type designed to fit into this range has a relatively crude and unattractive appearance (by typographic standards) even though it may be more pleasing to some than so-called unit fonts.[2]

```
This is a line typed on a standard typewriter.

This is a line typed on a proportional typewriter.
```

### 3. Lines of Fixed Length

Hot metal composition (and the setting of type from blocks of wood or metal, assembled by hand in the manner that preceded "machine" composition) requires that all lines be of equal length within a given area of type composition, such as a page. This is not to say that they must have the same *apparent* length, but each line must be filled out with spaces of non-printing height so that all lines (consisting of type characters plus spaces) will be precisely the same length. Individual pieces of type, or slugs from linecasting machines, can thus be assembled into a chase or form and locked up. The pieces of type would not stand on their feet and would work loose unless all areas within the form were filled solid and squared off, or otherwise subjected to equal tension and pressure from all sides. The type form, which looks like a metal picture frame with no back, must be capable of being lifted without the type dropping out.

The need to fill up a line does not exist with impact composition. This need does not exist, either, with other forms of cold composition, where *images* are dealt with rather than physical pieces of type, but the traditions of hot metal still prevail, so that many, if not most of the standards for cold composition are derived from the physical requirements of hot type.

Because lines of type composition in hot metal had to be of equal length the practice developed of "justifying" individual lines of type. The tradition emerged centuries ago, even before machine composition, that if lines had to be of the same length this condition should be made visually apparent, by applying the extra spaces necessary to equalize line lengths *between the words,* so that the end of each line would consist of a visual image rather than a non-printing space. We may therefore distinguish between ragged setting and justified setting as follows:

in *ragged setting* the spaces between words are uniform and the extra space needed to fill out the line is applied at the left, or at the right, or both;

in *justified setting* the spaces between words are of whatever width is necessary to provide left and right margins which are visually aligned. The practice is to have all spaces within a given line of equal width (or as close to the same width as possible so that differences are not discernible) but the space values in one line will not necessarily (and only coincidentally) equal those in other lines of the same text. Occasionally, to achieve desired effects or to solve the problem of intransigeant combinations of words, added space may be inserted not only between words, but also between letters. This is known as *letterspacing.*

[2] A font is a collection of type images of one general size and design.

Justified setting therefore implies variable interword spacing.

```
Justified setting can be achieved
on a conventional typewriter, but
to do this requires awkward vari-
ations  in interword spaces since
the   space  increments  are  not
small  enough to permit equaliza-
tion.
```

```
Justified setting can be achieved with
greater aesthetic satisfaction on a
typewriter with proportional type
characters, especially because there
are several different space values
possible.
```

### 4. How Lines are Justified in Hand-Set Type

A hand compositor assembles his type characters in a composing stick which is adjusted to a particular line length or "measure." First he fills the line, approximately, so that there is no longer room for another word or logical portion of a word. Then he goes back and adds space or takes out small increments of space between words, until the line fits tightly.

### 5. How Lines are Justified in Hot Metal (Linecasting)

There are two methods of setting type in hot metal machine composition. In line- or slug-casting (Linotype or Intertype), justification is achieved in the following manner. The operator keys a space by striking his spacebar. This causes a wedge-shaped spaceband to position itself between the type matrices which end one word and begin another. When the desired number of words in a line has been assembled, the space bands are driven up mechanically (to an approximately equal distance) sufficient to tighten up the line, and the line is then cast as a solid slug from these matrices, which are held apart by the spacebands. The spacebar thus inserts spacebands which create *justifying spaces.*

It is interesting to note that the individual type characters, although of varying widths, need not be designed so that there is any fixed relationship between the width of one character and the width of another (except for numbers, which may have to be arrayed in a tabular fashion and therefore are of uniform width). The type designer is free to express himself as he chooses in determining the aesthetic widths of individual characters. The spacebands will take up the slack in a line, and the operator can judge the number of characters to place in a line by visual inspection as each matrix falls into the assembler area. If he tries to set too many characters he may remove a few matrices by hand, but his eye can quickly learn to judge when a line is sufficiently full that it will cast properly without excessive interword space. The operator must consider not only the apparent width of the string of matrices, but will take into account the number of spacebands, since they are expandable.

For each line, therefore, he must make an *end-of-line* decision, observing whether the line will or will not justify. If it appears too loose he will add a word, or a portion of a word, if that word can be divided. His end-of-line decision therefore involves two considerations: *justification* and *hyphenation.*

### 6. Monotype Composition

The other method of hot metal typesetting is by Monotype. Here the operation is somewhat more complex because it involves two pieces of equipment: a *keyboard* and a *caster.*

The operator at the keyboard cannot see the type he is "setting." When he strikes a key his action perforates holes in a roll of paper tape. As he strikes a key the width of the character he has "set" is recorded or indicated mechanically by causing a wheel to advance a certain horizontal distance. The keyboard must therefore "know" the width of each character and it is necessary for the type designer to express character widths in conveniently additive values. An 18 unit system was developed, so that all type characters are described with an 18 in the denominator and some other lesser value (usually not less than four nor more than 18) in the numerator.

When the operator strikes his spacebar the mechanism assigns a provisional value to the spaces between words and tallies the number of such spaces. The justifying potential of the line (unless letterspacing is also to be used) is a function of the number of justifying spaces within the line. If only one appears, it is obvious that the expansion potential of that line cannot be very great. If there are five, or six, or more, the line is much more flexible.

By observing his counting mechanism, and looking for the appropriate warning signals, the keyboard operator can determine when he is within justification range, and he then terminates the line, looking at the justifying scale and keying at the end of the line the values required to achieve justification. The paper tape he creates is then fed into the casting machine with its end first, thereby conveying the necessary data for the casting of the spaces in that line before the line itself is set. It is also possible to cause the caster to insert minute amounts of "side-bearing" space, between characters within the word, which is a form of letterspacing throughout an entire line.

Considering the fact that the Monotype system was developed in the 1880s, it was a surprisingly advanced conceptual approach.

Both linecasting and Monotype composition will be discussed in greater depth later in this study. The only relevance at the present moment is to point out what justification is and how it is achieved, as well as to point out similarities and differences between typewriter composition and hot metal setting. So far the following observations have been suggested by our exposition:

1. when physical type characters can be seen (as in the instance of the composing stick or the assembler of the linecasting machine) no counting mechanism is necessary, especially where small increments of space can be added manually or mechanically.
2. when keyboarding is separated from casting (or character imaging) then in order to achieve justification some kind of counting arrangement is essential. This implies that type character widths must be

designed around a common denominator. Also that a count of interword spaces must be provided.

3. proportional type is superior, or is thought to be superior, to type of unit widths. Such type is even more difficult to "count up" than type of uniform character widths.
4. to achieve good justification small increments of space must be available to add between words.
5. impact composition cannot be justified without retyping or the use of some mechanical or electronic arrangement to capture keystrokes and play them back again with proper spacing added.

There are still other differences between typewriter composition and typesetting which need to be considered, but first let us try to summarize or check our understanding of what has been covered so far.

**Review Questions**

1. What is "cold" composition? What is "hot" type?
2. What is "impact" composition?
3. When is a typescript "composition" and when is it "copy"?
4. What is the relationship between "escapement" and character widths?
5. What do we mean when we say a machine must "know" the width of type characters?
6. What would you say would be the difference in readability between words made up from characters of uniform width and words made up from characters of proportional widths?
7. Which kind of type characters take up the most space?
8. What is the physical problem of assembling type characters into lines and locking them up in a form for typesetting?
9. Is the problem the same for cold composition as for hot? Why or why not?
10. What do we mean by "justification"?
11. How would you define ragged setting?
12. What is the relationship between justification and hyphenation?
13. What is a "justifying" space?
14. How is justification achieved on a linecasting machine?
15. How is justification achieved on a Monotype keyboard and caster?
16. What are the design limitations on Monotype type that do not exist for linecasting matrices?
17. What special problems are posed by the separation of the keyboard from the caster in Monotype?
18. What is "letterspacing"?
19. What relationship does the number of interword spaces in a line have to do with the quality of justification?
20. What is an "end-of-line" decision?

CHAPTER II

# Further Comparisons Between the Typewriter and the Typesetter

## 1. Recapitulation

We have seen that the typewriter produces cold composition by impact. Usually its character widths are uniform, but not invariably. Usually its product is ragged setting, and when an effort is made to make the right-hand margin look "justified" the interword spacing is awkward because the spacing increments are not small enough. If you want to justify your lines you may do so, by retyping, in the following manner:

```
Here is the measure I want to set to.

Now I begin to type and watch to seex
how close I can come to approximating
the measure I have chosen. In thisxxx
case I am placing an "x" at the endxx
of the line to indicate the number of
spaces I must add. I will add them at
the beginning of the line, from right
to left.

Now  I begin to type and watch to see
how close I can come to approximating
the  measure  I  have chosen. In this
case  I  am placing an "x" at the end
of the line to indicate the number of
spaces I must add. I will add them at
the beginning of the line, from right
to left.
```

We quickly learn to recognize that the standards of good typesetting owe a great debt to the conventions developed initially by hand composition, and imitated by hot metal machine composition. Later composing devices, especially the "cold" type methods of photo-composition and electronic character generation, which we shall be devoting so much time to, were called upon, for competitive reasons, to offer typesetting solutions which hopefully could achieve the same general aesthetic results as in the case of hot metal machine composition.

A typist is not likely to appreciate some of these fine points, until they are made obvious. One of them has to do with the greater readability of proportional type. Another has to do with the manner in which comprehension can be enhanced by the use of typographic emphasis. On a typewriter all you can do is to capitalize or under-

score. Perhaps some letterspacing would be possible. You might say, read this c a r e - f u l l y, and add letterspacing for emphasis to your stock-in-trade of typographic tricks. But in *typesetting,* and by this we mean *printers'* typesetting, you can use italics and bold face to lend emphasis. And you can also change type faces and styles. Different type sizes are available to you, and characters may be placed above or below the line for added clarity. All of these tricks cannot be produced by magic. They are often expensive. Sometimes they require a great deal of hand work, but they are features to what we shall call, somewhat snobbishly, *printers'* typesetting, which lend authenticity, persuasiveness, clarity, cogency and beauty to the written word. As we shall see, one of the most remarkable features of computerized composition is the tremendous flexibility it affords in this respect—far more than anything that can be achieved through conventional printers' typesetting.

We shall have occasion, shortly, to consider some of these typographic features. For the moment, however, let us think about some of the other differences between the typewriter and printers' type.

### 2. The Character Set

The character set for the typewriter is what the repertoire or array of characters is that will be found on the keyboard. It usually consists of 44 keys, plus a spacebar, a "shift" key, a backspace feature and some tab stops. Since the shift key implements upper and lower case aspects of the keyboard, it can be said that, for character repertoire purposes, the conventional typewriter offers a total of 88 characters plus a space. For the English language certain of these characters are fixed: there are 26 "alpha" characters, and in their upper and lower case variants this takes up 52 out of the available 88 plus space. There are ten numbers, and the uppercase positions for these ten numbers are more or less arbitrarily assigned. There is no "standard" keyboard layout in this respect. The author's typewriter has the following:

| Downshift | Upshift |
|---|---|
| 1 | ! |
| 2 | @ |
| 3 | # |
| 4 | $ |
| 5 | % |
| 6 | ¢ |
| 7 | & |
| 8 | * |
| 9 | ( |
| 0 | ) |

In addition, there are these keys and symbols:

| | |
|---|---|
| - (hyphen) | _ (underscore) |
| = | + |
| ½ | ¼ |
| ; | : |
| ' | " |
| , | , |
| . | . |
| / | ? |

How do these compare with printers' type? First of all, we should remember that some typewriters economize in the use of keys by asking the user to keyboard a lower case l (ell) for a 1 (one). On our typewriter there are two keys, but only one image. In printers' type it is extremely important to be able to differentiate these two, and one would think this should be the case on a typewriter, as well, but it seldom is.

Turning to the "non-alpha" characters—numbers, punctuation marks and the like—perhaps the first need is for what is known as an "em dash"—a horizontal dash longer than a hyphen—which is usually simulated in typewriter script by the conjunction of two hyphens. In addition, most typesetters would agree that a "nut dash"—a horizontal dash longer than a hyphen but shorter than an em dash—is also highly desirable. It is used to connect two dates, such as 1888-1960, or to join compound strings, such as "real-time-on-line device." Attention should also be paid to the problem of quotation marks. There are "left quotes" and "right quotes" and single and double quotes. Usually a conversation opens with two left single quotes, which makes up one double left quote, and closes with two right single quotes. One right quote by itself is also used as an apostrophe.

Some of the other characters, especially above the numerals, are not so essential for a standard layout. There never will be enough room on a standard typewriter keyboard for all of the characters frequently used, and fractions invariably present a problem which will require more extended discussion. But attention should be called to the fact that many good type faces and fonts—used commonly for book composition—contain combinations of characters known as *ligatures,* where two or more letters are joined into one. These are: ff, fi, fl, ffi and ffl, and for them there is of course no typewriter equivalent.

Why make a comparison between printers' type and the typewriter keyboard? For two reasons: first because the printer has the problem of converting the typewritten manuscript into type. Second, because the typewriter keyboard, with certain modifications, is used increasingly as an input device for computer typesetting. It is therefore necessary to bridge the intellectual gap between the typewriter and type. As we shall see, there is a third reason, too, to be concerned about the typewriter-printers' type relationship: that is the possibility of reading typewriter-produced manuscripts by optical character recognition (OCR) devices, and creating printers' type from the signals so generated. This leads us to a consideration of certain other typographic peculiarities. So far we have not been directly concerned with problems of *format.*

### 3. Considerations of Format

Certain format conditions are so obvious that we do not stop to think about them. Consider, for example, the problem of *spaces.* It is sufficient to have only one kind of space key on a typewriter. We do not need to be concerned about the difference between a fixed space and a variable or justifying space. But printers must know the difference. A paragraph indention is always a fixed space, usually an "em" space. But there are also "en" ("nut") and thin spaces on linecasting machines. A nut space has the value of a single digit numeral and is often used to achieve alignment of numerals. A thin space may be used simply to set two characters or symbols slightly apart by a constant amount. Here are examples of how these spaces might be employed:

This is the beginning of a paragraph.
□ ●1.↑This is part of a list.
□ 10↑This is another part of the same list.

□ indicates an em space.
● indicates an en or nut space.
↑ indicates a thin space.

Another format problem is to be able to recognize the end of a paragraph. This is what printers call a "quad left" condition. This means, in hot metal, that the rest of that line must be filled out with "quads"—non-printing fixed spaces which will bring the line to the desired measure. On a linecasting machine the keyboard operator will strike his keys a sufficient number of times to produce the required amount of space, or else a device known as an "automatic quadder" will do this for him. On a Monotype keyboard the operator will fill out his line with fixed spaces. An output signal to show a paragraph ending is not needed. The new paragraph will be indented or otherwise separated from the preceding paragraph. But an input signal, for typesetting purposes, is very much required for computer typesetting. At this point all we want to establish is that there are certain format conditions which typists take for granted, but which typesetters must think about.

Many of these conditions are more easily solved on a typewriter than in printers' type. Proportional type causes serious problems of alignment. It is not too difficult to set a table on a typewriter because of the unitary value of all characters and spaces. It is much more difficult to do this in type, and if you rely on variable-width spacebands to justify your lines it may be impossible.

### 4. Quadded Lines vs. Running Text

Of course it is only running text that is justified—until you encounter the last line of the paragraph, which is quadded (although if it almost takes up a full line it may be spaced out).

Quadded lines are set with a constant amount of interword space, and they are generally set "flush left," "flush right," or "center." Other locations can be achieved by including fixed space or indentions in the measure before the left, right, or centered decision is taken.

Here are some illustrations. The lines which are quadded are flagged for easy identification with a QL, QR or QC. Note that the fact of quadding means that *a line does not run over* (since only the portion which "turned over" would then be quadded, and the balance would be justified). The excess space is placed to right, left, or both, *rather than between the words.*

| | |
|---|---|
| Chapter I | QC |
| *A Face at the Window* | QC |
| Primary Considerations | QL |
| 1. Wash thoroughly | QL |
| 2. Rinse until free of soap | QL |
| 'Twas brillig and the slithy toves | QL |
| Did gyre and gimble in the wabe. | QL |

Table I QC

Figure 1 QC

Caption QR

### 5. Flush Left and Right and Leadering

Another format condition is the positioning of material on a given line to the extreme left *and also* to the extreme right, with all of the extra space in the middle.

| | |
|---|---|
| Cleveland | Ohio |
| Portland | Oregon |
| Seattle | Washington |

Or, leaders, which lead the eye from left to right, may be used to fill in the space:

Cleveland . . . . . Ohio
Portland . . . . . Oregon
Seattle. . . . . Washington

Just as these effects are difficult to create on a typewriter, so are they difficult to achieve in type—sometimes much more so, due to the proportionality of the type. As we shall see, computer programs are sometimes able to perform certain of these tasks more consistently and more economically than human beings.

### 6. Identification of Problems and Situations

The most difficult task for a person entering the field of computer typesetting is to define the problems to be faced and the solutions desired. There are problems of *system, equipment,* and *typographic requirements.* We have endeavored, at the outset, to suggest some of the differences between typewriter composition and typesetting in order to create an awareness of some of the unique aspects of the challenge. We are concerned with a peculiar blending of art, craftsmanship, technology and aesthetics. We shall not be useful in this field unless we understand much about all of these things.

**Review Questions**

1. What do we mean by "character set" or "character repertoire"?
2. What typographical characters are missing on a typewriter that are widely used in typesetting?
3. When is an "em dash" used? When is a "nut dash" used?
4. List four kinds of spaces used in typesetting.
5. What does "quadding" mean? What is the derivation of the term?
6. Why isn't it necessary to draw a distinction between a quadded line and a justified line for typewriter composition?
7. What is "leadering"?
8. Why is it important to recognize the difference between typing requirements and those of typesetting?
9. How can you convey emphasis with typewriter type? How do you do it with printers' type?

CHAPTER III

# Characteristics of Type

Although nowadays much typesetting is done by means of photo-composing devices, it is still necessary to think of type as if it were made up of individual blocks of metal, each containing a raised letter which carries ink. For the most part type characters are still designed as if they were to be cast of metal, and it is useful to know what this implies in order to appreciate the ways in which type images can differ from their conventional prototypes when they are composed from film negatives or electronic character generating devices.

A typographic image has always been designed as if it sat on a rectangle of white space, with a "base line" some distance from the bottom of the white space, depending upon the length of the descenders and ascenders of the face in question. Although of course the type was designed as a mirror image, if we ignore this fact for the moment we can visualize words made up of pieces of type to appear somewhat as follows:

It will be observed that the letters "y", "p", and "g" descend below the base line. Also, notice that each character has its own unique width. A differently-designed type face could have different relationships:

Here we have two designs of letters which take up the same white space (vertical height) but have different widths, different base lines, different descenders and different "x" or "z" heights.

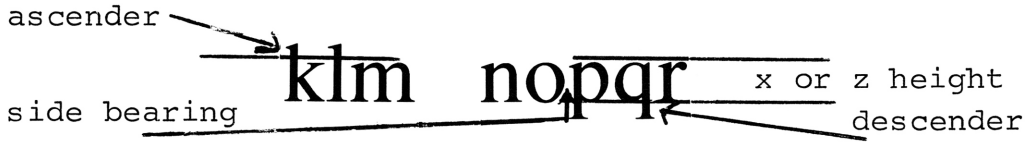
Here are some useful terms:

"x" or "z" height—the height above the base line of a lower case character with no ascender.

ascender—portion of lower case character that goes above the x height

descender—portion of lower case character that goes below the base line

side bearings—the non-printing area (in hot metal that portion of the metal) which is found to the left and right of the character itself.




**1. Type Measurement**

We speak of type as "eight point" or perhaps "ten point" type. We mean by this that the vertical white space the character takes up is eight points or ten points high. The statement tells us nothing about the horizontal dimension of the character or the space it occupies because each character has its own unique width, depending upon its design, such as Times Roman or Baskerville. By looking at the printed image it is even difficult to identify the point size unless we know type very well since some faces are "larger" than others (i.e., they have a larger x height and shorter ascenders and descenders).

One way to try to determine the point size is to measure the distance, in running text, from one base line to the next, but this can be misleading since the custom has developed of "leading" (pronounced "ledding") lines of type—that is, inserting strips of metal between lines to set them further apart, or to cast them as if such strips of metal were positioned below them. Thus we often speak of "ten on twelve", by which we mean a ten point type face cast onto a vertical body of twelve points, or with two extra points below the apparent base line. This separates the ascenders and descenders so that they do not come so close together. Especially for "wide measures" (long lines) it is easier for the eye to find the next line when it returns to the left of the column.

pppp pppp gggg gggg yyyy yyyy qqqq qqqq
TTTTT TTTTT TTTTT TTTTT TTTTT

pppp pppp gggg gggg yyyy yyyy qqqq qqqq
TTTTT TTTTT TTTTT TTTTT TTTTT

And so we have observed that different type faces in the same point size have different base lines, different heights and different relative widths. And these lines may or may not be additionally separated by "lead."

Some type faces are more compact than others. Type characters of a given point size may take up less overall width. This will of course affect the number of characters that will fit in a given line. Even the same type face may be designed in a "condensed" or an "expanded" version. It is not easy, therefore, without knowing a great deal about type, to identify a type size by looking at the width of individual characters.

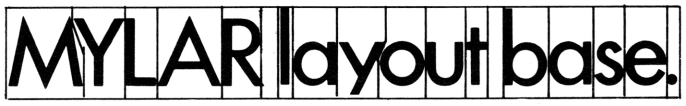
There is a unit of measurement called the "em." This is a theoretical area which is supposed to be the square of the point size. (In fact, because of "leading" it is more relevant to think of it as a unit of width rather than of height.) However, the concept is that a twelve point em would be twelve points high and twelve points wide. It usually just accommodates the capital letter "M", but not always. In a few type designs the letter "M" may be wider or less wide than the "em" space. But if they are the same width, the letter "M" would fit on an em space somewhat as follows:

and so it can be seen that the actual letter is neither twelve points wide nor twelve points high. The actual height will be seven or eight points, and its width perhaps nine or ten points. (Just to confuse things further, it sometimes happens that the type designer, in creating an extended or a condensed face, will proportion it to an "em" space which is actually conceptually wider or narrower than it is high.)

### 2. Measuring Individual Characters

Type characters may be designed to be expressed as increments of an em space, or they may not. Foundry type is generally not so designed, unless it is copied after a Monotype face. But it was necessary for Monotype to be designed so that each character's width could be expressed as a fraction of an 18th of an em, including side bearings. An "i" might be 5/18ths, a capital A might be 10/18ths, and so on. This imposed a serious limitation on the type designer, and yet even with this handicap some of the most beautiful type in the world conforms to the proportions of 18. On the other hand, some linecasting type, which did not have to meet this requirement, and afforded greater flexibility to the designer for this reason, was nevertheless unattractive. As we shall see, linecasting imposes another kind of physical limitation—the matrix. Whenever two characters are positioned on the same matrix ("up" and "down" the "rail") the second character must take on the width of the first, including its side bearings.

Because of the rectangularity of metal type, one character cannot lie over or under another. It is not possible to set type in metal so that the following effects are created (unless the type is "mortised," by cutting away some of the side bearings in an irregular fashion):




Where more than one character is cast from the same matrix (linecasting "up" and "down" the "rail") the second character must be arranged or designed so that it has the same width as the first. On a Monotype grid an italic character (say an "a") can have a

different width from a roman one. But with linecasting, the width of the two a's, including side bearings, must be the same. Since italic characters slant, and should also be placed closer together than roman characters, linecasting italics have never been very handsome.

In metal typesetting each size has to be produced from different molds or matrices. While two or three sizes in a given range may be designed after the same pattern, it is customary to redesign the patterns for larger or smaller ranges because the proportions which seem valid for one range do not seem satisfactory for another. (Say that a range is a spread perhaps from six to nine points). In general, the larger the type the more delicate, in proportion, are the hair lines and serifs, and the higher the character in relation to its width. A photographic enlargement of a ten point character, doubling its size (increasing its area by 4x) would make it appear smaller and cruder than the same character redesigned as a twenty-point character. Thus one objection to photocomposition is that some machines produce larger or smaller point sizes merely by photographic enlargement or reduction. Not only is the image less satisfactory, but a character "fit" that looks good in one size will not look nearly so satisfactory in another.

As we shall soon learn, some phototypesetting devices (first generation phototypesetters) were designed on the basis of hot metal typesetting principles, and these cannot offer improved typographic flexibility. The type images are still encased in metal molds or matrices. But all second and third "generation" photographic typesetting machines, which produce images from negatives, can implement type designs which will position one character so that it overlaps another. This can be done in two ways: either by the design of the type itself, or by a special facility called "kerning." If all characters would look well in overlapping positions—that is, if a "y" could "tuck under" any character which precedes it, then the "y" can be so designed. But if only certain combinations of characters will permit this (as a V following an A, but *not* following an N) then the kerning effect must be achieved either by a special keyboarding solution (negative partial escapement) or by a computer program whose purpose is to accomplish the same result.

### 3. Type Faces for Photocomposition

There are two methods of creating type images for photocomposition. One is from a photographic negative and the other is, in effect, the photographing of a character which has been generated on the face of a CRT (cathode ray tube—like a TV tube) by electronic means.

When the type character is produced from a photographic master, the negative image is projected through a lens system onto photo-sensitive film or paper. The character image master is invariably a negative. The typeset character is invariably a positive (although film can be treated by a reversal process to produce a negative image).

The individual type character must be "laid down" or positioned at the proper place along the line of text, and so, just as the typesetting keyboard has to "know" the width of the character, so the photosetting device must also "know" how much to "escape" before, or after, it exposes the character.[1] In first generation text photosetting machines (as distinguished from devices built to produce display type only, by semi-manual

[1] Some phototypesetters flash and then escape, while others escape and then flash.

methods) the widths and escapement values are dealt with in much the same way that they are in hot metal—by the width of the brass matrix (in the case of the Fotosetter) or by the position on the grid (in the case of the Monophoto). But in second generation phototypesetting devices—which were constructed on different principles from hot metal machines—other methods of "knowing" escapement values came into being. These will be described subsequently, but to illustrate one principle: when the code for the character is "read" by the machine, the machine simultaneously accesses its own memory bank, which tells it how far to move a travelling prism or lens along a given path in order to focus the character at the proper location.

It should be evident, then, that a photographic image can, in principle, be positioned anywhere, if the proper signal can be produced—that there need be no physical barrier to such positioning. It is even possible to position one character directly over another one, as is done with "floating" accents, for example.

Since phototypesetting devices involve the use of lens systems, a larger or smaller final image can be created by magnification or reduction, but not only will the size of the character be affected; so may its *position* with respect to the base line. For example, the Linofilm uses only one lens, which moves forward or back to enlarge or reduce the size of a character. Hence the base line (if different sizes are intermixed on the same line) would look as follows:

A A A

But if discrete lenses are used for each magnification or reduction, as is true in the case of the Photon machines, for example, the magnification or reduction need not affect the position of the base line.

N N N

### 4. Treatment of Leading

Just as the horizontal widths of photo images are generally controlled by photographic or electronic means[1] as distinguished from physical or mechanical ones, so there are also no longer physical limitations that bear upon the "leading" of the line. The distance of one line of type from another is usually controlled by a "film advance" mechanism which moves the photosensitive film or paper, by rollers or sprockets, so that the next line can be exposed. All images are generated on the same vertical plane and the exposed portion of the film is then re-wound into a light-tight cassette.[2]

[1] Except for the first generation Fotosetter and Monophoto.

[2] This is not true of the Linotron 1010, which exposes an entire page at one time.

If, therefore, the type face is defined as "ten on twelve," the film would advance twelve points between base lines. In general, in photocomposition, the leading factor is *above* the line, not below it as is the case with hot metal. The film advances twelve points, and thus provides the new base line. This means that it is now necessary to think somewhat differently about type. Instead of visualizing descenders as part of the type area, in a sense they extend *beyond* the type area. In other words, in a base-line oriented system, the type area looks like this:

Here is one line of type
Here is another line of type

Not this

Here is one line of type
Here is another line of type

Consequently, if you move from a ten point face (on twelve leading) to a six point face (on six leading), extra space must be provided for at the point of transition, or else the result will look like this:

This line is ten on twelve.
This line is ten on twelve.
This ppppp line is ten on twelve.
This line is six on six.
This line is six on six.

This, of course, affects copy markup for typesetting. The person who performs mark-up has to know the difference between hot metal treatment of vertical space and photocomposition treatment of it.

### 5. Points and Picas

We have been referring to "ten point" or "twelve point" type without explaining what it is. And we have used the terms "ems" and "ens" without relating ems to points. It is important for people working with type to understand the theory of the point system, and so we will take some pains to explain it.

A *pica* is a typesetter's measure. Pressmen are not very familiar with picas. They think in terms of inches. But typesetters think in picas. A pica is roughly one-sixth of an inch. There are roughly six picas to the inch. Actually, six picas (rounded off) are .996 inch.

The pica is used in English-speaking countries. On the continent the unit of measurement is the Didot point or the Didot-Berthold System. The Didot point is slightly larger than the American. We will confine our discussion to the pica system. A pica is divided into points, of which there are twelve to the pica, and therefore approximately 72 to the inch.

Picas and points are *absolute* measurements, whereas ems and ens are *relative.* An em of ten point type is ten points wide, and an en is five points wide.

An em of twelve point type is of course the same as a pica. *Two* six-point ems equal one pica. Three eight-point ems equal two picas.

Three ten points ems = 3 x 10 = 30
$^{30}/_{12}$ = 2 $^{6}/_{12}$ or 2 picas and six points.

### 6. Conventional Type Sizes

Type has customarily been designed in point sizes such as 5, 6, 7, 8. . . . Not all type faces are designed in all sizes. Certain electronic typesetting machines (CRT typesetters) have the ability to create character images in half point sizes. They also have the ability to advance film (i.e., handle leading) in fractions of a point (e.g., one-tenth or 1/32nd of a point). Consequently it is now possible to ask for a book to be typeset in 9 ½ points on 11.2 leading.

### 7. The Concept of "Set"

In the past, the term "set" or "set size" has been used to indicate the amount of white space that might surround a character. In Monotype a character can not only be set on a base whose vertical height is, say, two points larger than the body the type was designed for (ten on twelve)—which merely means that the lines are farther apart—but also the character can be cast on a base whose horizontal dimension is also larger, even though the image of the character itself would remain precisely the same size.

The character does not change its shape when it is cast on a wider or taller base. It is only the white space which changes—the vertical leading or the "side bearings."

But now a new concept of "set" has been introduced with electronic composition. This is a form of character distortion. It is possible to generate the character on the face of the CRT tube as if it is from a 10 point font so far as height is concerned, but from a 9½, 10½ or even 18 point font so far as width is concerned. The side bearings are not changed. They remain the same proportion to the width of the character as they were before. But the character itself changes.

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 12 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 11.5 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 11 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 10.5 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 10 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 9.5 set**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**abcdefghijklmnopqrstuvwxyz — 12 pt., 9 set**

Now, in view of this new capability, it is most convenient to think of type design in terms of three elements:

Point size will refer to the height of the character. A ten point character will be about six or seven points high, but designed in relation to a family of characters that will use up a ten point dimension of vertical space (allowing for a little room above ascenders and below descenders).

Set size will refer to the width of the character. A ten set character will be whatever proportion of a ten point em that character was designed to be.

Leading will refer to the distance between one base line and the next.

**Review Questions**

1. In printers' type, would a Times Roman "i" and a Baskerville "i", both in ten point, be the same width?
   a.) Yes, necessarily.
   b.) No, not necessarily.
2. Would a ten point em space be the same width in Times Roman and Baskerville?
   a.) Yes, generally speaking, by definition.
   b.) No reason for them to be.
3. Would a ten point em space in Times Roman be the same width as a nine point em space in Times Roman?
   a.) Yes, since they are both em spaces.
   b.) No.
4. Base lines of hot metal type always align, regardless of type design. (T) (F)
5. With a few exceptions, base lines of photocomposition align, regardless of size and design. (T) (F)
6. Leading is additional white space (above) (below) the line.
7. How does the concept of "set" differ between Monotype and electronic CRT typesetting?
8. "Kerning" is impossible in hot metal setting without mortising. (T) (F)

CHAPTER IV

# Typesetting Devices that Antedate the Computer

In order to understand the implications of the computer revolution in the typesetting industry we need to know something about the technology as it existed prior to the application of the computer. It is logical to start with linecasting equipment.

## 1. Characteristics of Linecasting Machines

There are two manufacturers of linecasting equipment: Mergenthaler ("Linotype") and Harris ("Intertype"). Both machine lines are similar in their basic principles and can therefore be discussed as if they were one product. The Linotype was invented in the 1880s by Ottmar Mergenthaler and has undergone evolutionary developments since that date, but still operates according to the same basic concepts.

The standard linecasting keyboard is not laid out like a typewriter. Each key represents a "channel," and depressing the key causes a matrix to fall from its position in a "magazine" or storage box, and find its way to the assembly elevator. Each matrix has two intaglio images on the reverse edge, and a representation of those images on the forward edge, so that the operator can identify the characters as they fall into place. Of the two images on each matrix, the one to be cast is determined by the position of the "rail," which is set manually by the operator. When a word or phrase is to be set in italic, for example, he will position the matrix on the rail, by hand or by lever so that the alternate intaglio image will be cast.

After the matrices have been assembled, the spacebands push up to fill the line (they can be expanded to about three times their smallest width), and the slug is cast. The matrices are then elevated and deposited in the distribution mechanism, and they find their way back to their home channels. This is accomplished with the asistance of a unique configuration of teeth in the top of the matrix, which is recognized by the distribution bar.

The length of the measure to be set is determined, within the limits of the capacity of the machine, by the manual adjustment of the jaws in the assembly. (One modern model has the capacity to adjust its jaws automatically in response to a signal.) The usual line width can be up to 30 picas, but some machines can set measures up to 42 picas.

Magazines, containing type matrices, have mostly ninety channels. There can also be auxiliary magazines of 34 matrices, and a side keyboard to access these additional positions.

Linecasting machines in most common use access only one magazine at a time, and since 90 channels or 90 distinct matrices are available, each with two intaglio images, 180 different type characters can be set in a given measure at one setting. Different sizes can be obtained by mounting a new magazine. This must be done manually. (The magazine, if made of brass, weighs approximately 70 pounds).

Additional magazines can be mounted and held in reserve on some models, so that several are available to the operator without much physical effort on his part. Certain models of machines permit mixing of type from different magazines within the same line, and other models only permit moving from one magazine to another for the setting of independent blocks of copy.

Type sizes can range from four points to sixty points, but generally speaking linecasting machines are used to set "text" material, in ranges from eight to fourteen points.

## 2. TTS Linecasting

In the 1930s a method was developed to drive linecasting machines by means of punched paper tape. Such tape was already in use for the transmission of teletype messages. A modified tape format was developed for wire service transmission of news stories, so that lines which were pre-justified at the news bureau could be received on tape and fed directly into linecasting equipment, without the necessity for each receiving station to perform a rekeyboarding operation. However, this method of typesetting came to be used not only for the composition of information relayed by wire, but also for many in-plant applications.

A TTS keyboard, with a counting mechanism, enabled a keyboard operator, physically separated from the linecaster, to make end-of-line decisions. The keyboards used at that time, and still widely used today, were "blind." That is, they provided no "hard copy." No visual record of the keyboarder's keystrokes is provided. The only indication that the keyboarder has available is the inspection of the configuration of holes in the paper tape itself, and the ability to turn back the tape and "rub out" unwanted keystrokes by superimposing delete codes.

In order to make his end-of-line decisions the keyboarder watches an indicator scale which keeps track of the cumulative widths of the characters already recorded within the line, as well as the number of space bands. When the cumulative width counter and the spaceband counter meet, the line is considered to be within justification range, and the keyboarder can strike the keys which provide the end of line codes.

The punched paper tape is then mounted on the linecaster's paper tape reader, and the codes, which are detected by mechanical fingers which generate electric pulses, activate an adapter keyboard mounted under the manual keyboard, and this causes the appropriate keys to be depressed just as if they were being manipulated directly by an operator. (Some linecasting machines were subsequently built to run *only* from paper tape).

It is obvious that the counting or justifying keyboard had to have some method of "knowing" the width of each character as it is called for, so that the running total can be cumulated. The first step, therefore, was to design new type faces, all of universal widths. In other words, linecasting matrices were redesigned so that every "a" had the same relative width, usually expressed in terms of a constant fraction of a 32nd of an em.

A multiface perforator was then developed, which permitted the insertion of a "counting magazine" or code bar so that it was no longer necessary for a given character to be assigned the same relative width, regardless of type design. It was only necessary to insert the appropriate counting bar into the keyboard. It even became possible for typesetters to make use of their conventional linecasting faces, which were not designed according to a common denominator (other than in terms of thousandths of an inch), in view of the fact that spaceband expansion could usually cope with discrepancies in character widths, where these widths were rounded to the nearest 32nd of an em.

A TTS keyboard layout is somewhat comparable to that of a typewriter. (The term "qwert" is sometimes used to describe the standard typewriter layout, since this is the sequence of the first five alphabetic characters at top left). However, the following additional keys are present:

QL
QC
QR
UR - upper rail
LR - lower rail
Em
En
Thin
Return
Elevate
Bell or Stop
Rub-out

Small caps and ligatures may be obtained by utilizing certain combinations in the "shift" and "rail" positions.

TTS paper tape is a six level advanced sprocket tape with 63 binary code combinations. The diagram on the following two pages illustrates the manner in which the characters are accessed. It is not a complete listing, but is shown for illustration only.

It will be observed that certain codes can stand for as many as four different characters, depending upon whether they are in one of the following conditions:

UR or LR
Shift or Unshift

In a sense, then, the reading of a "precedence code" which signals "shift" INDICATES THAT UNTIL FURTHER NOTICE ALL CHARACTERS ARE IN THE SHIFT MODE. THE "UNSHIFT" key returns the condition to the lower case mode. Similarly, the "UR" *code signals that until further notice all characters will be in that mode, and the "LR"* code signals out of that mode and back into the normal mode. The same code therefore can operate in four modes:

| | |
|---|---|
| LR Shift A | UR Shift *A* |
| LR Unshift a | UR Unshift *a* |

Within LR you can move from SHIFT to unshift.
Within unshift you can *move to UR* and LR.

| Characters Accessed in "Shift" Position | | Characters Accessed in "Unshift" Position | | Bit Configuration |
|---|---|---|---|---|
| UR | LR | UR | LR | |
| A | A | a | a | 011 000 |
| B | B | b | b | 010 011 |
| C | C | c | c | 001 110 |
| D | D | d | d | 010 010 |
| E | E | e | e | 010 000 |
| F | F | f | f | 010 110 |
| G | G | g | g | 001 011 |
| H | H | h | h | 000 101 |
| I | I | i | i | 001 100 |
| J | J | j | j | 011 010 |
| K | K | k | k | 011 101 |
| L | L | l | l | 001 001 |
| M | M | m | m | 000 111 |
| N | N | n | n | 000 110 |
| O | O | o | o | 000 011 |
| P | P | p | p | 001 101 |
| Q | Q | q | q | 011 101 |
| R | R | r | r | 001 010 |
| S | S | s | s | 010 100 |
| T | T | t | t | 000 001 |
| U | U | u | u | 101 000 |
| V | V | v | v | 001 111 |
| W | W | w | w | 011 001 |
| X | X | x | x | 010 111 |
| Y | Y | y | y | 010 101 |
| Z | Z | z | z | 010 001 |
| scV | 1 | scW | ffi | 110 111 |
| scB | 2 | scM | ffl | 111 001 |
| scC | 3 | scA | ? | 110 000 |
| scD | 4 | scO | * | 101 010 |
| scE | 5 | scL | ff | 101 000 |
| scF | 6 | scK | ae | 110 101 |
| scG | 7 | scN | | 111 100 |
| scR | 8 | | - | 101 100 |
| scT | 9 | scF | fi | 100 011 |
| scJ | 0 | scY | fl | 101 101 |
| scS | : | | ; | 101 011 |
| scT | ( | scQ | ) | 110 001 |
| scI | sc& | | & | 101 001 |
| | | SPACE | | 000 100 |
| | | SHIFT | | 011 011 |
| | | UNSHIFT | | 011 111 |
| | | UR | | 110 011 |
| | | LR | | 111 011 |
| | | EM | | 110 100 |
| | | EN | | 101 110 |
| | | THIN | | 100 000 |
| | | TAPE FEED | | 000 000 |
| | | DELETE | | 111 111 |

| Characters Accessed in "Shift" Position | | Characters Accessed in "Unshift" Position | | Bit Configuration |
|---|---|---|---|---|
| UR | LR | UR | LR | |
| | | open quote | closed quote | 100 010 |
| | | period | period | 100 111 |
| scH | oe | hyph | hyph | 110 010 |
| | | QUAD LEFT | | 110 110 |
| | | QUAD RIGHT | | 111 110 |
| | | QUAD CENTER | | 101 111 |
| | | BELL OR STOP | | 111 010 |
| | | CARRIAGE RETURN | | 000 010 |

Note: Functions shown in CAPS are valid in shift, unshift, UR or LR

Presentation could have been as follows:

| Characters Accessed in UR Position | | Characters Accessed in LR Position | | Bit Configuration |
|---|---|---|---|---|
| Shift | Unshift | Shift | Unshift | |
| A | a | A | a | 011 000 |

etc.

Further note: Some TTS tapes are more conveniently read from reverse side. In this case reverse bit configuration, read from right to left.

TTS Tape is "Advance Sprocket" to indicate direction of tape. Tape moves forward in direction indicated by sprocket position.

000.000
010.111 ↓ Period indicates position of sprocket hole. It aligns with forward edge of bit holes and moves in that direction.

### 3. What are the Advantages or Limitations of TTS?

With the separation of keyboarding from the setting or casting function there are certain inherent gains, but there are also certain limitations.

1. The TTS keyboards resemble more closely the conventional typewriter layout, and it is easier to obtain and train personnel, thereby helping to resolve a possible manpower shortage problem.
2. The keyboards are faster to operate than a linecasting machine keyboard since the cycle for the latter includes the elevation and distribution of the matrices and the casting of the metal, both of which tend to limit the rate at which lines can be processed. Moreover, the operator's visual inspection of assembled matrices probably deters his keyboarding efficiency, as compared to the lesser distraction of watching the justifying counter on the TTS keyboard.
3. Keyboarders can be employed during the day, when they are more readily available, and machine casting can take place over three shifts, thereby providing for more intensive utilization of the more expensive capital equipment.
4. On the other hand, greater productivity is required to break even. If one machinist monitors three linecasters on a given shift, and there are three keyboard operators, all being paid as much as linecasting machine operators or monitors, and if productivity were no greater than straight keyboarding at the linecasting machine, wage costs would have increased because four persons rather than three would now be engaged. Consequently, the breakeven point must require greater productivity from the *three* keyboarders than was formerly achieved by *four* machine operators. At least 25% more keystrokes must be achieved, and considering the capital investment, this should run closer to 30 or 35% for breakeven.
5. If one machinist monitors only two linecasters, the breakeven ratio must, of course, be higher.
6. Only a limited class of work can be handled by TTS. Work involving the frequent use of pi mats is more difficult to produce, although it is possible to load a pi channel with pi mats in the sequence in which they will be expected to fall. This takes more job planning, and if one fails to fall in the right sequence it would require more corrections.
7. The error rate for TTS setting tends to be higher and this places a greater burden on the proofreading and correction cycle. The rate is higher because keyboarders may make more errors on blind keyboards (although this is debatable) but also because of malfunction of the linecasting machine in reading and operating under tape instructions.

It is important to recognize, however, that TTS setting offers an alternative to computer composition. The author suggests that many of the advantages which have been attributed to computers (in hot metal setting) could have been secured at lower cost by the installation of TTS equipment.

### 4. TTS Setting Affords the Transition to Computer Typesetting

Had it not been for the development of TTS it is possible that computers would not have been applied so readily to the task of typesetting. But TTS tape provided digitized input for computer processing, and tape readers on linecasters provided a means of final output. All that was necessary, therefore, was to supply the intermediate step whereby the computer takes over the function of counting character widths and spaces and making end-of-line decisions.

### 5. Hot Metal Setting by Monotype

When TTS was introduced, typesetting by linecasting became somewhat more similar to typesetting by Monotype. Monotype had always involved the separation of keyboard from caster, and had always used paper tape input.

In many respects, Monotype would have been more congenial to computerization, since the width values on the basis of which its type was designed had always been reckoned on a common denominator. However, the paper tape that drives a Monotype caster was not computer compatible, and there was, therefore, a serious problem of how to adopt the Monotype system to the computer age.

A Monotype paper tape spool is approximately four inches wide, and there are positions to accommodate 31 channels across one line of the tape. The tape is sprocketed on both sides so that it advances, both in punching and in reading, by a line at a time.

The original Monotype system used a square matrix case on which could be positioned 225 characters. There are 15 horizontal and 15 vertical rows, designated as 1 through 15 and A through O. The coordinates of a given position on the mat case can thus be described as A 14, for example. On the tape a hole would appear in one location only, designating A through N (with no hole for O) and a second hole would appear for 1 through 14, with no hole for 15. This is the sequence across the tape:

```
                                                  1 1 1 1 1
M L K J I H G F / E D / C B A 1 2 3 4 5 6 7 8 9 0 1 2 3 4 /
              Space  .0075                                .0005
```

The holes in the paper tape permit compressed air to move into certain channels which cause the square mat case to be positioned over the type mold, so that the intaglio character on the mat case can be cast, with its desired body, as a single character. Thus the mat case continually jumps from location to location as it responds to commands from the paper tape.

It may be seen, then, that the coding on the tape is in no sense "binary." The tape might well be compared to that on a piano player roll, where the position of a hole across a horizontal line determines what note is to be played. But here one position governs a vertical coordinate, and the other a horizontal one.

Within the 15 x 15 mat case, characters are positioned according to their widths, and there needs to be no pairing of UR and LR equivalents, since each character can have its own unique location, so long as its width can be described within the Monotype frame of reference. The top row will consist, usually, of characters which are 5/18ths of an em in width; the next row, those which are 6/18ths. The next, 7; the next, 8. There would probably be three rows of 9 units since all the numbers would be this width, along with certain other characters. Next would be two rows of 10s, perhaps, and then one of 11, one of 12, one of 13, one of 14, one of 15 and one of 18. However, the allocation of width values to rows and the assignment of characters to rows will be determined by the design of the type face in question. Different code bars are inserted into the Monotype keyboard so that a particular combination of holes will be punched depending upon the matrix layout, and the appropriate width values will be accumulated by means of the counting mechanism, and indicated on the justifying scale.

The counting scale will show the "deficit" which must be added to the tape to cause the line to justify. Justification can be accomplished in several ways, but principally by causing interword spaces to vary according to the positioning of the .0075 and .0005 wedges, which are activated by commands punched at the end of the line.

As we have said, the main difficulty in computerizing the Monotype system arose not from the principles of machine operation, which are thoroughly congenial to computer typesetting concepts, but to the problem of hardware adaptation. No computer-activated punch existed for the preparation of a 31 channel paper tape. Two approaches were therefore taken. One was to modify the paper tape tower of the caster so that it could be activated by solenoids which opened and closed air passages upon command from a six level paper tape (binary) reader—a device built by P. Lennart Andersson upon order from Rocappi, Inc.—and the other was to devise a converter which read six level paper tape and punched out a 31 level Monotype tape off-line. This latter approach was more widely used, since it required no modification of the caster, and was first applied commercially in England by Rocappi, Ltd., in cooperation with the British Monotype Company.

Subsequently, other solutions were developed, and there is now available an entire system for computerized Monotype or Monophoto composition. On the whole it can be said, however, that the most significant strides in the field of computerized composition, while owing a great debt to the Monotype unit system, owed still more to the concepts and technology of TTS.

### 6. Impact Typesetting Devices

At least a very brief mention should be made of tape-driven impact composition devices, since their development also made a contribution to computer typesetting technology.

Typewriters have long been used as composition devices, despite the obvious limitations of non-proportional type and limited interword spacing characteristics. When proportional type was applied to typewriters there was still the problem of typing the line twice—once to find out how short it would be (from the desired measure) and again to add the needed extra interword spacing. This led to paper-tape driven typewriters. One impact device which is still widely used, especially for suburban newspaper composition, is the Friden Justowriter. The Justowriter consists of two units—a Recorder and a Reproducer. The Recorder provides a punched paper tape and a hard copy for proofing. The keyboard operator sets a counter to indicate the desired measure, and then types across the line until a warning light appears. The operator then strikes a carriage return code and the appropriate justifying information is recorded. The punched tape is then mounted on the Reproducer unit for automatic retyping in proportionally-spaced and justified lines. Each Reproducer is limited to one size and one type face, as determined by the characters on its striking bars. Proportionality of the type is limited to a narrower range of width alternatives than would be the case for hot metal composition. (Much of this same technology was incorporated into an early phototypesetting device made by ATF, in models known as the A, the B and the B8.)

The code structure of the Justowriter is basically six level in character, with two additional levels being added for parity or justification codes. Corrections can be made in the paper tape by mounting it on the Reproducer unit, if equipped with a tape punch head, allowing it to type out and simultaneously punch out a new tape up to the point where the line correction is desired. The operator can then by-pass the unwanted line and type a new line which will be recorded on the new tape.

### 7. The Advent of Photocomposition

In the 1950s photocomposition devices for text typesetting began to come into the market. In response to what needs were they developed?

First of all, the growth of offset printing made it possible to use photographic images of type characters at least as advantageously as type itself. If the type were set by hot metal processes it was necessary to pull a "repro" proof in order to use it as camera copy.

Offset printing from plates made by photography from repro copy also permitted the use of alternate methods of page makeup. While repro proofs could be produced from locked-up pages, pages could also be assembled by stripping and paste makeup techniques, so that it was no longer necessary to incorporate furniture and strips of lead, and to quad out lines, in order to lock up the type in a frame. Since the objective was to obtain positive or negative film, any technique which accomplished this was permissible. Justification was no longer necessary for purposes of makeup and lockup, although it was still desirable for reasons of appearance, especially since offset was trying to compete for the same markets with letterpress. But the absence of the lock-up requirement meant that white space (where no type images appeared) could be provided merely by the arrangement and positioning of the material which was to be reproduced. Lines did not have to be quadded out. Space above and below heads could be provided merely by the way in which the ingredients within the copy were located.

We shall find out, subsequently, that there are circumstances in which hand paste-up or stripping, to space out material, is more expensive and less precise than the use of lead and furniture, but there are also many circumstances in which the opposite is true. Therefore, one of the major advantages of photocomposition is its ability to handle white space, merely by non-exposure, in certain areas, of the photographic film or paper.

Another parallel advantage of photocomposition lies in its ability to avoid the hand assembly of the various ingredients which had previously had to be set by different machines or by different methods. In hot metal typesetting a manuscript is "cut" or divided into its different type elements. One operator sets the text matter, another the extract, a third, the footnotes, a fourth the display heads. Then all of these elements have to be assembled and placed in position. If the photocomposition device is capable of offering a wider range of type faces and sizes, much of the hand makeup can be eliminated during the process of typesetting, thus avoiding not only the hand assembly but also the copy cutting and subsequent pulling together of the manuscript copy.

Not all phototypesetting devices have this versatility, but many of them do.

Another advantage of phototypesetting is its ability (in many machines) to access a much greater character repertoire than is offered either by linecasters or even by Monotype.

Finally, there is also the advantage of speed in composition. Hot metal typesetting is limited by the necessity of cooling metal after it has been cast. Something on the order of perhaps fifteen newspaper lines per minute is about as fast as hot metal typesetting devices will ever go. Perhaps by adding refrigeration or freezing technology the level of output could be doubled, but it could never be made to equal the speed of a photocomposing device which might achieve 40, 100 or even 1000 newspaper lines per minute. And so we can list the potential advantages of photocomposition as:

1. flexibility in the treatment of white space, formats and measures
2. greater versatility with respect to type sizes and the character repertoire itself
3. speed

On the other hand, the output of the photocomposition device is more difficult to correct. And the potential speed of photocomposition requires new input solutions, for with the greater capacity of the devices to output type comes the need to find ways to simplify the input. It is the computer, happily, which has offered the solution to many of the problems which held back the development of photocomposition. Computers can handle corrections. They can simplify problems of input, and they can offer solutions to makeup in those areas where, otherwise, hot metal spacing out of pages is more easily performed than the manual stripping of material.

In the next two chapters we will examine the characteristics of principal types of photocomposing machines, and then we will begin to consider how computers can provide new solutions, in terms of *typesetting systems.*

**Review Questions**

1. How do linecasting machines differ from Monotype machines?
2. Why must the width of the upper rail character correspond with the width of the lower rail character positioned on the same matrix?
3. What limitations exist with respect to width values for Monotype that do not exist for linecasting?
4. What limitations exist with respect to width values for linecasting that do not exist for Monotype?
5. What is the character repertoire of the typical linecasting machine?
6. How can you change measure in setting with linecasting equipment?
7. How can you change type size in setting with linecasting equipment?
8. Explain how TTS works.
9. How does a TTS keyboard differ from a linecasting machine keyboard? How would these two differ from a Monotype keyboard? (The latter is not explained in this chapter. Try to find out what you can do about it.)
10. How can 64 six level codes be made to convey machine function commands and also access as many as 180 different type characters?
11. How are small caps and ligatures keyboarded on the average linecasting machine? How are they keyboarded on a TTS keyboard?
12. What are the advantages of photographic composition as compared with hot metal? What are the limitations as compared with hot metal?

CHAPTER V

# Photocomposition Devices

We shall continue our discussion of typesetting devices that antedate the computer, and in this chapter we shall discuss some of the principles underlying the design and operation of the first two "generations" of phototypesetting equipment. There are many makes and models of such equipment and we cannot hope to cover them all. Nor would such a catalog serve a useful purpose for an introductory work such as this. Our concern should be conceptual, so that the reader is able quickly to understand the underlying theory of phototypesetting operation, and the evolutionary trends with respect to equipment development.

While reading this account, the student should have in mind three basic questions, and one supplementary one. These are:

1. How is the character image generated?
2. How does the photo unit "know" where to "lay down" the character and how much escapement or width to assign to it?
3. How does the machine perform justification, or does it?
4. What are the special features of the machine, in terms of:
   a. character repertoire
   b. range of point sizes
   c. quality of image
   d. length of measure
   e. speed of the machine
   f. whether or not it is a computer slave—whether it can be driven by an independent keyboard; whether it makes "its own" end-of-line decisions
   g. other special features, such as reverse leading, base alignment, etc.

Unfortunately, we shall not have the space to cover many of the points listed above. The reader who wishes to research the topic in depth, perhaps because he is considering acquiring such a device, may, however, be guided by these questions.

## 1. First Generation Machines

Two typesetting machines evolved from their hot metal counterparts. A third evolved from an impact typesetting device.

The *Harris Intertype Fotosetter,* which was developed immediately after World War II, and introduced to the industry in 1950, is constructed much like a linecasting machine. It retains the principle of the circulating matrix. Matrices are released from a magazine, find their way to the assembly elevator, and after the line is composed they are sent

back to their channels in the customary way, by means of elevator, identifying teeth and a distributor bar. Each matrix contains a photographic image of one character, which is positioned in the center of the matrix, rather than on its reverse edge. A light source exposes the character onto film or paper. The width of the matrix controls the escapement of the image across the measure. Spacebands are not used. Early models distributed the "deficit"—the amount by which the line is short—between characters as well as words, but later models successfully achieved the allocation of the extra space between words only.

The Fotosetter is manually operated from a conventionally-arranged linecaster keyboard, and it offers four magazines, each with 117 channels. But only one character is positioned on a matrix, rather than the two "duplexed" characters which are usually associated with an upper and lower rail. A double distributor permits mixing of 228 characters. Extra ("pi") mats can be inserted into the line by hand, just as an operator can supplement his character repertoire in hot metal linecasting by the manual addition of "sorts."

Matrices of 6, 8, 12 and 18 points are available, and a lens system, with fourteen different lenses, operating manually by means of a revolving turret, offers size ranges from 3 to 72 points.

A TTS, tape-operated machine, called a *Fotomatic* was also manufactured, beginning in 1962. One 90-channel magazine, with duplexed mats, provides the character images. Eight different sizes can be set from each Fotomat. The speed is approximately 20 newspaper lines per minute.

* * *

Just as a Fotosetter resembles a linecaster, so a *Monophoto* resembles a Monotype caster. Punched tape is prepared in the same manner as for Monotype. But the "caster" is a photo unit. Negative images are mounted in a mat case (you can put together your own combinations), originally of a 16 x 17 design, but with a 17 x 20 configuration on the "Mark 4."

A moving mirror positions characters across the line. Justification is controlled by tape in much the same manner as with Monotype. Manual adjustment of a single lens will reproduce different sizes of images. The machine exposes at the rate of two or three characters per second.

* * *

The *ATF Typesetter* runs from punched tape, just as the Justowriter does. It resembles the Justowriter in appearance, except that type characters are mounted on a revolving disk, which contains 168 images. The disk shifts to the indicated position *and then stops* while the character is exposed. With other devices which store their character images on disks, the image is flashed onto film or paper while the disk continues to move. Moreover, with other such devices the image is stepped across the line by means of a shuttle-like motion of a lens or prism. But with the ATF the image is exposed in a stationary position, and it is the carriage that moves. Escapement is accomplished by a gear mechanism which controls the movement of the carriage. The carriage contains the cassette into which the film of photographic paper is inserted. The machine "knows" the number of units in each character so that it can determine how far the carriage should shift before it "lays down" the next exposure. There is no magnification or reduction of image size. For larger sizes of type a new disk is mounted, and the gear ratio is changed manually, since a "unit" of width now requires a larger movement of the carriage.

The speed of this machine is also in the neighborhood of three characters per second.

## 2. Second Generation Photocomposition Devices

It may be convenient to think of "second generation" devices as those which are not adaptations of early typesetters, but which were conceived of initially for purposes of setting type photographically from type character masters (almost invariably negatives). Such machines differ radically in design from the first generation concepts, contains fewer moving parts and more electronic circuitry.

The first two such machines to come to the market were the *Photon 200* and the *Mergenthaler Linofilm.* The Photon 200 is manually operated from a keyboard (although it was subsequently "automated" to run from tape), whereas the Linofilm consists of two units—keyboard and photounit—which are physically separated. From now on in our discussion, all of the devices that we shall describe, except the Photon 200, are based upon the principle of separating keyboard input from character generation.

The *Photon 200* consists, then, of a keyboard and photo unit under the control of a keyboard operator. The keyboard resembles a conventional typewriter, mounted on a desk with many supplementary controls. Buttons are provided for point size, set size, type face, leading, and line length. A line counter keeps track of the cumulative widths of the material set. Until the line is composed it is held in a buffer area. Then it is released to the photo unit while the operator begins to compose the next line.

The photo unit houses a disk which contains 16 fonts of 90 characters each, each taking up a separate concentric circle around the disk. The disk is first positioned so that the font desired is centered before the light source and lens system. Then, when a character is to be set, a flash exposure, synchronized for the position of the desired character, projects the image through the lens system and stepping prism onto the photographic film or paper. The operating speed, when lines are being composed, is about 10 characters per second. Of course the photo unit only operates when the keyboarder has given it a line to set.

A lens turret, with 12 different sized lenses, makes available a combination of 12 sizes x 16 fonts x 90 characters, for a total of 17,280 different combinations. (Later Photon models even offer multiple disks!). This is perhaps the largest character repertoire of any second generation photo unit. Escapement is controlled by plug boards which are consulted by the photo unit as well as by the counting mechanism of the keyboard. The correct plug boards must be mounted for the character widths of that particular disk.

Later models of the Photon 200 separated the photo unit from the keyboard. This presented some problems, because it then became necessary to devise keyboards which could carry sufficient information on all the width values of all the desired fonts to assist the operator in making his end-of-line judgments.

Other Photon models can be operated from paper tape created from separate keyboards. There are three types which need to be mentioned. Each uses a different method of conveying information about width values to the photo unit. All present the same keyboarding problem: how to access character widths and generate enough codes to send commands for font shifts and lens changes.

The Photon 513 requires the insertion of plug boards. A separate plug board must be mounted in the photo unit for each of the 16 fonts on a disk. (Actually, there is room only for 15 plug boards, so that two fonts must have the same relative width values if all 16 are to be used at once.) The Photon 540 and 560 *carry their width information to the photo unit* on the input paper tape. In the case of one of the machines, three six level codes are used for each character. In the case of the other, two eight level rows are used.

Considering the latter, as an example, the first row of eight levels identifies the beginning of a new character and tells what character it is. The second byte tells the photo unit what the width value of the character is to be.

We have seen, then, that a photo unit can "know" its width values by consulting plug boards inserted for the fonts in question, or by being "told" on a character by character basis, on the input paper tape. What other method of conveying width value information is used?

The Photon 713 is a more sophisticated machine, electronically, than the 200 and 500 series, and it contains a small computer memory which serves two purposes:

1. the sequence of characters can be identified and their codes given:
2. width values can be stored.

The various models of the Photon 713 operate not on the principle of a revolving disk, with a light source behind it, but of a revolving drum, with a light source inside of it. Using the Photon 713-10 as our example, on the drum will be mounted two segments of film strips. On each film strip there are two duplexed rows of characters, making up four fonts. The rows are duplexed because the same width values must apply for the characters in each row. Characteristically, roman and italic are duplexed, and this means that they must be designed to conform to the same basic width standards which have always applied to hot metal type, where two intaglios are duplexed on the same matrix. Considering the two film strips, the following diagram illustrates the principle:

| | | |
|---|---|---|
| Light Source A | Film Strip No. 1 | |
| | Top two rows— | Times Roman duplexed with Times Italic |
| | Bottom two rows— | Times Bold duplexed with Times Bold Italic |
| Light Source B | Film Strip No. 2 | |
| | Top two rows— | Univers duplexed with Univers Italic |
| | Bottom two rows— | Univers Bold duplexed with Univers Bold Italic |

Light source *A*, when it flashes its stroboscopic light, will illuminate *all four* characters which happen to be positioned in front of the light source and lens system at that moment. A shield or mask arrangement which moves vertically between the drum and the collimating lens permits only one row of the four to be selected at one time and thus passes along one character image through the lens system (there are eight lenses on a turret, for eight different sizes of type), and to a moving prism. If an image is desired from film strip No. 2 the illumination will come from Light source *B*.

Returning now to the method of loading information into the memory bank of the 713, a paper tape is prepared and a switch is set to indicate to the paper tape reader and photo unit controller that the information on that tape is to be loaded into memory, and is to serve as code identification information. This tape is then read, and each eight-bit byte is loaded consecutively, so that the code structure for each byte will correspond with the sequence of characters around the drum. Command codes are also identified and stored. In effect, then, this makes it possible for the photo unit to accept any eight-level code configuration.

After the code identifiers have been loaded, another switch is set which indicates that width values will now be loaded for the widths of the duplexed characters for film strip No. 1, top duplexed fonts, and a paper tape containing these width values will load these codes into memory. The machine now knows that the character in the first position around the drum is identified by a certain code and has a certain width. The operator then loads width values for the other three duplexed fonts (the bottom two rows of film strip No. 1 and both duplexed rows for film strip No. 2).

Once the load tapes have been read in, the switch is changed to indicate that the next information to be received will consist of text input. As each pre-justified line is read the machine holds the line information in its buffer and computes the amount of interword space which must be assigned in order to cause the line to justify. In other words, the arithmetic unit in the Photon performs the same function as the spaceband in hot metal linecasting, but the input tape must contain all necessary codes to indicate which lens is to be used, which characters are to be set from which row on which film strip, and what other machine functions are desired (film advance for leading, for example).

It is not an easy matter to get all of these codes onto the tape in the first place, and most input keyboards are not sufficiently versatile to permit the full utilization of the photo unit's capabilities. If, however, straight text in two faces only, and of one size, is all that is desired, an ordinary TTS perforator can be used to create the input. The "up the rail" code can be used to give the command to access the duplexed (italic) font. If additional fonts are to be accessed then one or more additional codes would be required. Often a code is assigned (called a "supershift") to indicate that a second duplexed pair is to be accessed. But since the Photon 713 is capable of reading eight level tape, many other code commands can be given (for lens or size changes) and additional fonts, if they can only be keyboarded. Additional keyboard features are sometimes added (often by attaching a "black box") to an eight-level Friden Flexowriter keyboard or a similar keyboard device to provide more input capability. It can be seen that there are two types of problems: how to get additional set width information to the *keyboard* and how to generate additional *photo unit* command codes for lens and font changes.

The Photon 713, since it has a memory and small computer of its own, can also be modified to perform other functions. One capability which is offered is to permit the 713 to make "its own" end-of-line decisions. Since it must calculate width values in order to justify lines, adding to interword spaces whatever values are required, and since there is no physical limitation to the amount of interword space that can be assigned (such as must be the case for mechanical spacebands) the 713 can be told (in some models) to read a tape which does not contain end of line signals, and when it encounters a word which causes the line to overflow, to back up to the last space and then recompute the interword space values, breaking the line at that point. This causes the Photon 713 to perform a task known as "*hyphenless* justification," because its capabilities do not include those of breaking words when an entire word would not fit, but a portion thereof would. It follows, of course, that the quality of hyphenless justification, especially on narrow measure material, will be very poor. On long lines the interword space values may be quite acceptable.

Photon models provide base aligning fonts. In other words, since a lens turret is used to produce different sizes of type, these lenses can be so focused and adjusted that the same base line is used as the reference point regardless of the size of the image. However, this does not mean that when type is mixed in the same line there will be the same precise alignment, since even type of the same point size (necessitating no lens

change) may be designed with longer or shorter descenders and ascenders, and a different "x" height, and so the apparent base line may differ from one face to another. Such variations in base alignment can be avoided in third generation machines, but not in second generation ones, unless lens adjustment can be effected for each unique combination of type faces.

* * *

The *Mergenthaler Linofilm* consists of two principal units: a keyboard and a photographic unit. At the keyboard the operator has access to a typewriter (which produces a hard copy) and many auxiliary controls. He can set the desired minimum width of a space, he can select any one of eighteen different grids or fonts, and he can indicate which of six type sizes he wishes to produce from that grid. When he keys a character he can, if he wishes, add extra space to the value of that character, or subtract space from it. He is informed at all times precisely where he is in the line, since there is a continuous cumulative count of the values he has keyed, and by writing down these values he can return to the same place in the line in order to create special effects, such as multilevel mathematical equations.

The operator's output is recorded on fifteen level paper tape which is, unfortunately, not computer compatible. One row across that tape records not only the character identification, but also the width or escapement of the character in question. In other words, the tape conveys to the photo unit the desired width information on a character-by-character basis, and provides an opportunity to modify that escapement (to achieve kerning effects, for example), because width values are thus capable of modification.

A Linofilm photo unit contains a "grid turret" of eighteen square, glass-covered grids of 88 characters each. On command from the paper tape input, one of these grids is selected from its turret, swiveled into position before the lens and light source, and a character is exposed onto photographic film. There is only one lens, and the size of the character is therefore determined by two considerations: the size of the photographic master, and the position of the lens. There are three sizes of masters, and six lens positions. Thus 18 different fonts of one size master could be produced in six different image sizes, or six different fonts (each with three size ranges) could be mounted, and produced in six different sizes each. Because the lens enlarges or reduces by racking in or out, the characters set from different lens adjustments are not base aligning. Size mixing in the same line is difficult, but the quality of the photographic image, being generated from three size ranges, is superior to that created in the same range of sizes from only one photographic master.

The speed of the Linofilm on straight text setting is approximately 12 characters per second. However, there is considerable delay in making changes from one grid to another or from one size to another, even though these are performed in response to automatic commands. (Models produced after July 1965 are somewhat faster).

Paper tape input into the Linofilm photo unit is read by two different read heads. The first scans the tape to the line-end code in order to obtain the information necessary for it to perform the justification function (distributing the needed white space between words). The second read head interprets the commands for the setting of specific characters, and activates the actual character exposure routines.

The Linofilm has been made computer-compatible in three ways. Converters exist which will translate computer output, paper or magnetic tape, off line, into fifteen level paper tape. Mergenthaler also supplies a read head which will accept six-level paper tape, and store three rows of codes, providing the same information which would normally be obtained from one fifteen-level row. Or, it is possible to purchase a COL

model (Computer Output Linofilm) which reads a six level tape, does not perform any automatic justification, and responds solely to the information provided by the computer. These models offer 28 character grids of 88 characters each.

It is now possible for us to make a point about photo units which, although implied by our presentation to date, may have eluded the reader: photo units which were built to operate directly from keyboards must have some kind of logic (or some mechanical contrivance) built into them *to enable them to perform the function of justification.* The keyboard operator makes an end-of-line decision which *merely brings the line into justification range.* Just as spaceband expansion actually justifies the line, so must the photo unit contrive to expand its space values to achieve the same results. But if a computer is used to calculate width values it can also output specific codes to instruct the photo unit precisely how much space to put between words. If the photo unit ignores this information and performs its own independent calculations, then two different devices are performing the same task, and for one of them the task is unnecessary. Thus, when the computer output of a typesetting program goes to a Linofilm photo unit the tape can by-pass the first read head entirely, and every line, although justified, can be set exactly as a quad left line. Fixed spaces take the place of variable spaces, *but they are fixed spaces of variable widths,* all carefully calculated by the computer program. In fact, for Linofilm, it is possible to add the value of the interword space onto the value of the first character in the word following the interword space, since the Linofilm unit *escapes and then flashes,* and each character can carry its own width value with it, as it is input into the photo unit.

* * *

The *Harris Intertype Fototronic Filmsetter* also has a keyboard and a separate photographic unit. The keyboard produces an eight-level punched paper tape, and it has a continuous-counting mechanism which derives its width values from rotating disks which are mounted at the keyboard. The photo unit produces its character images from revolving disks (a five disc model is available; each disk contains 240 characters) and width values are recognized by the photo unit by its reading of a diagram or coded value on the disk itself, which conveys the escapement information. The paper tape which goes into the photo unit is read twice: once for end-of-line information, in order to perform justification, and once again to expose the characters called for. A ten point master will produce sizes from 5 to 36 points. A 15 point disk will provide a range from 7.5 points to 54. The optical system, which is quite complex, permits 19 sizes of type from one master.

## 3. Summary

Certain first and second-generation phototypesetting devices have been discussed in order to illustrate some of the problems which need to be considered. The review questions at the end of this chapter will serve to highlight the considerations which we feel to be important to an understanding of the fundamental principles involved.

In general, all of the approaches we have discussed in this chapter involve the generation of character images directly from photographic masters. In one case a same-sized image is all that is possible. In other cases, character sizes can be varied by the use of lens systems. Sometimes an entire gamut of sizes is produced from one master. In other cases a given master can be magnified or reduced within a fairly narrow range and more than one master image is necessary to embrace a full range of type sizes.

There are different ways of conveying information about escapement values. The first generation machines handle this problem mechanically. There are several different solutions so far as second generation equipment is concerned.

All of these devices perform the function of justification, trying in some way to imitate the function of the space band.

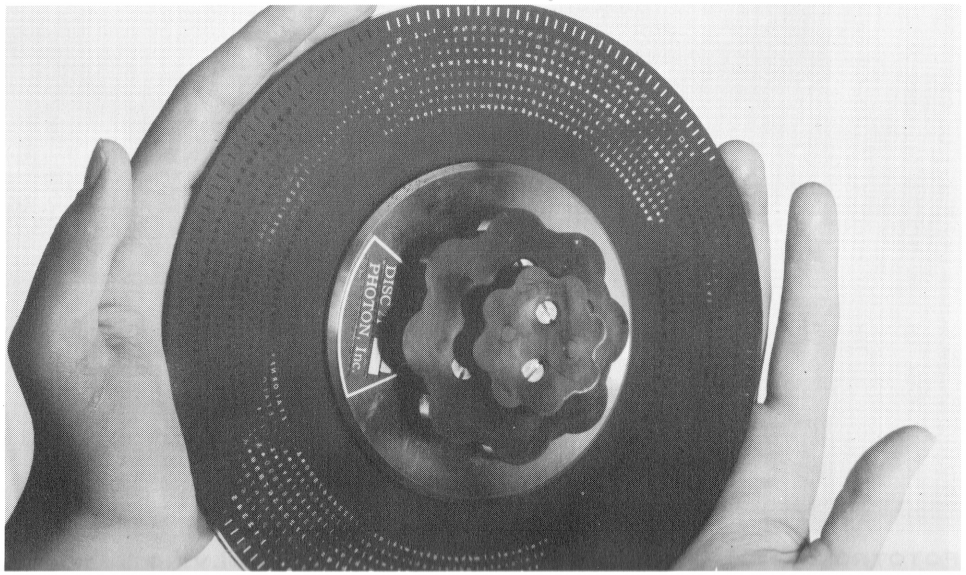
Most of these devices have separated the keyboard from the caster or photo unit. The latter functions more or less automatically, although some require considerable manual adjustment to "set up" the machine for certain sizes, and to mount or load film strips, disks and the like. Where the photo units are capable of mixing many sizes and fonts a very real problem is presented to the keyboarder. How does he find out what the width values are that he requires? How does he relay the necessary signals as to font changes, size changes, and the like, directly to the photo unit? Do the complexities of keyboard operation require a level of intelligence and efficiency, to say nothing of monitoring and button-pushing, which takes away the convenience of photo unit flexibility? Is it better, after all, to assemble different sizes of type by hand than to try to do it by means of keyboard control? There may be no easy answers to these questions.

**Review Questions**

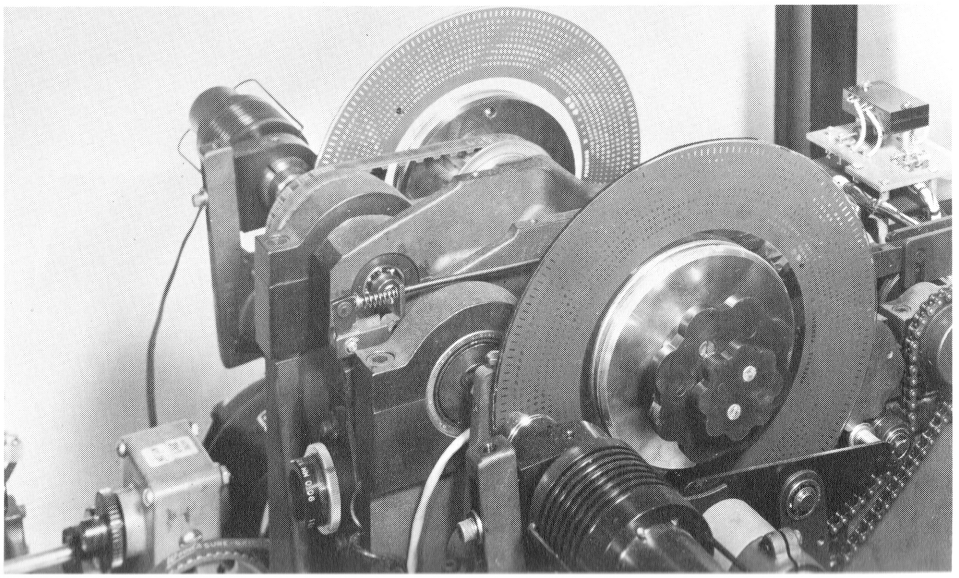
1. How would you distinguish between "first" and "second" generation photocomposition devices?
2. How is character width information conveyed to the camera of the Fotosetter? How is it conveyed to the Monophoto?
3. For those Photon typesetting machines which read paper tape, how is width information conveyed to the photo unit?
4. How does the Photon 713 differ from the Photon 500 and 200 series?
5. When paper tape is produced for a second-generation photounit, by a keyboard operator, how are lines justified? Is justification done by the operator or by the machine?
6. Which phototypesetting machine use revolving disks? What other methods of character storage are used and on what devices?
7. How is it possible to change the escapement value of a particular character to be set on a Linofilm photo unit when it cannot be done on a Photon?
8. What can you set on a Linofilm photo unit that you cannot set on a linecasting machine?
9. What do we mean by "duplexed" characters or fonts?
10. What is meant by a "supershift" code?

THE ORIGINAL FAIRCHILD JUSTIFYING KEYBOARD. NOTE JUSTIFYING SCALE, HOLE IN RIGHT SIDE FOR INSERTION OF WIDTH PLUG BOARD.

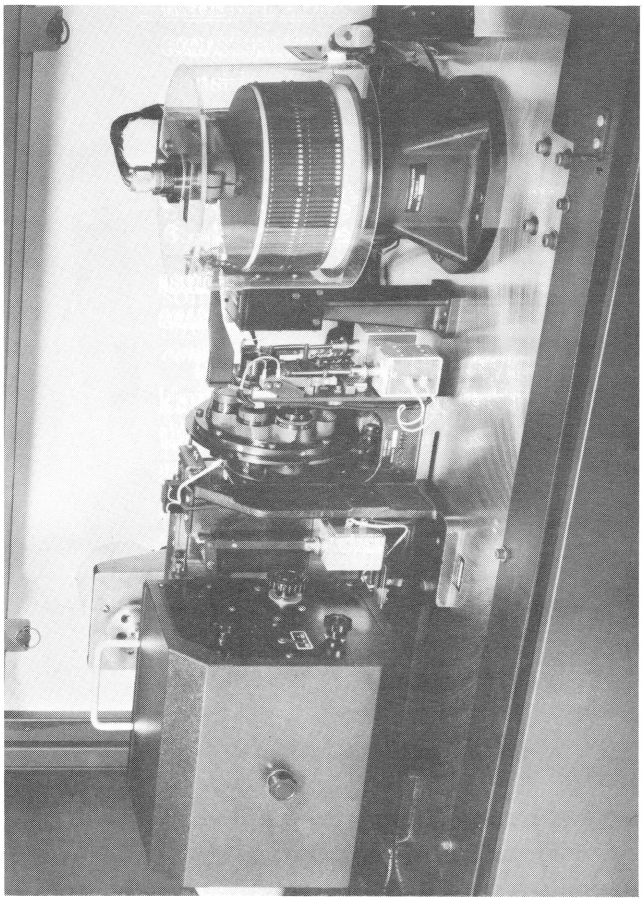
A FRIDEN JUSTOWRITER SYSTEM, USED TO PRODUCE JUSTIFIED, PROPORTIONALLY-SPACED LINES OF IMPACT COPY. THE JUSTOWRITER AUTOMATIC COMPOSING MACHINE IS A TRADEMARK OF THE SINGER COMPANY.



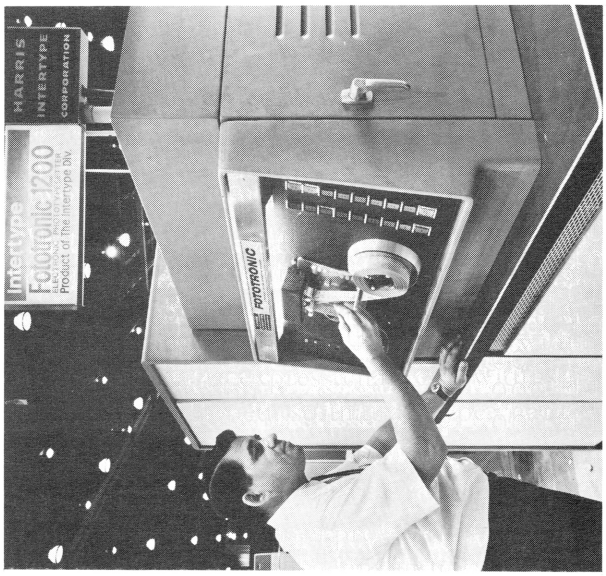

A TYPE DISK FROM PHOTON, SUCH AS IS USED ON THE 200 AND 500 MODELS.



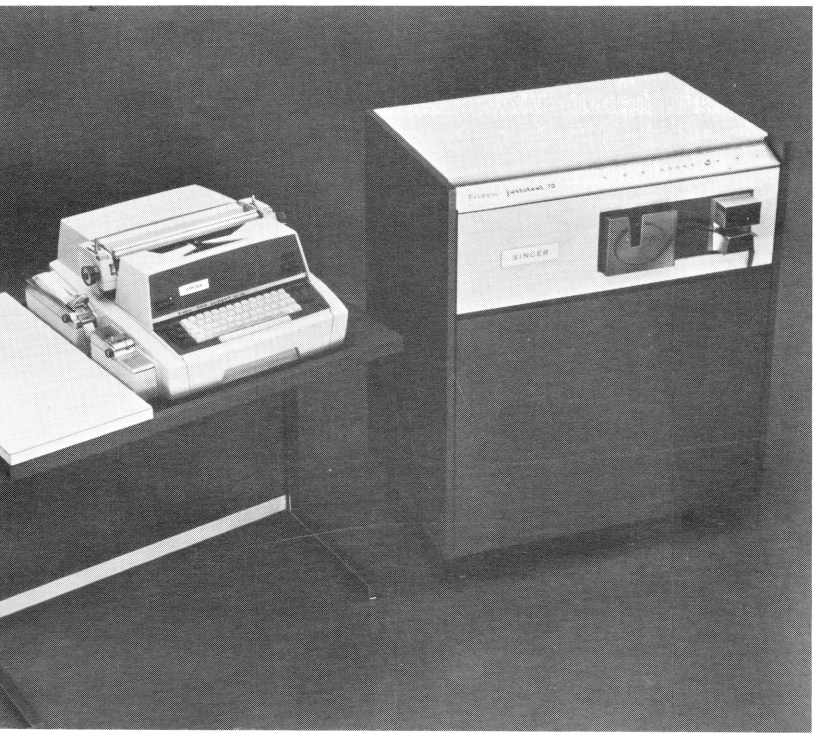
INSIDE SHOT OF THE PHOTON 532 FONTMASTER.



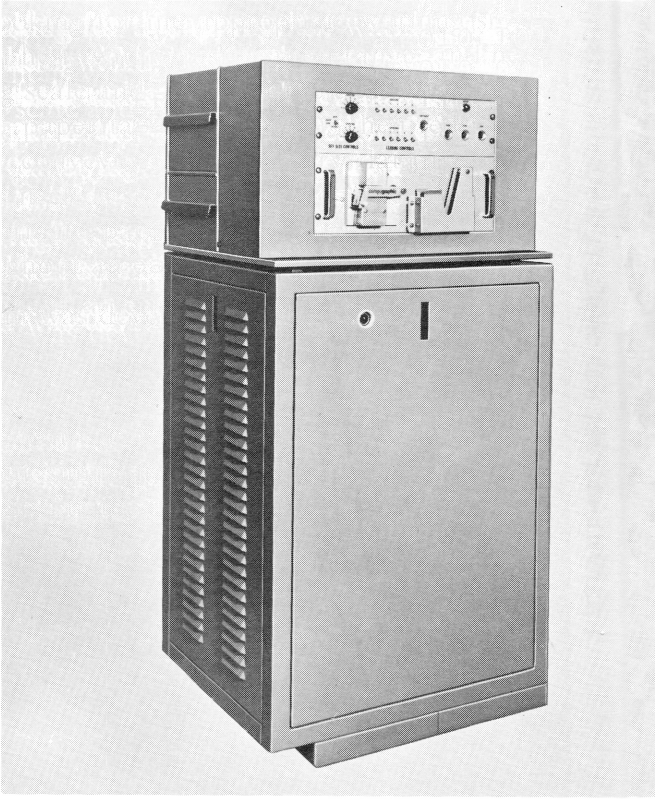
PHOTON 713–10, INSIDE SHOT SHOWING REVOLVING DRUM AND LENS SYSTEM.




FOTOTRONIC 1200, USING THE PRINCIPLE OF THE REVOLVING DISK, WITH FIVE SUCH DISKS.




FRIDEN JUSTOPERF* TAPE PERFORATOR (LEFT) AND JUSTOTEXT* 70 PHOTOTYPESETTER (RIGHT) BOTH PRODUCTS OF THE SINGER COMPANY.



COMPUGRAPHIC CG4962. THIS MACHINE WILL SET FROM RIGHT TO LEFT AND FROM LEFT TO RIGHT FROM A 2-FRAME 6-LEVEL COMPUTER-GENERATED PAPER TAPE.

CHAPTER VI

# "Third Generation" Typesetting Devices

According to the definition we shall invent, a "third generation" typesetter is one which:

a. is devised to accept input primarily from a computer rather than directly from a keyboard, *and*

b. generates its character images on the face of a CRT tube, *and*

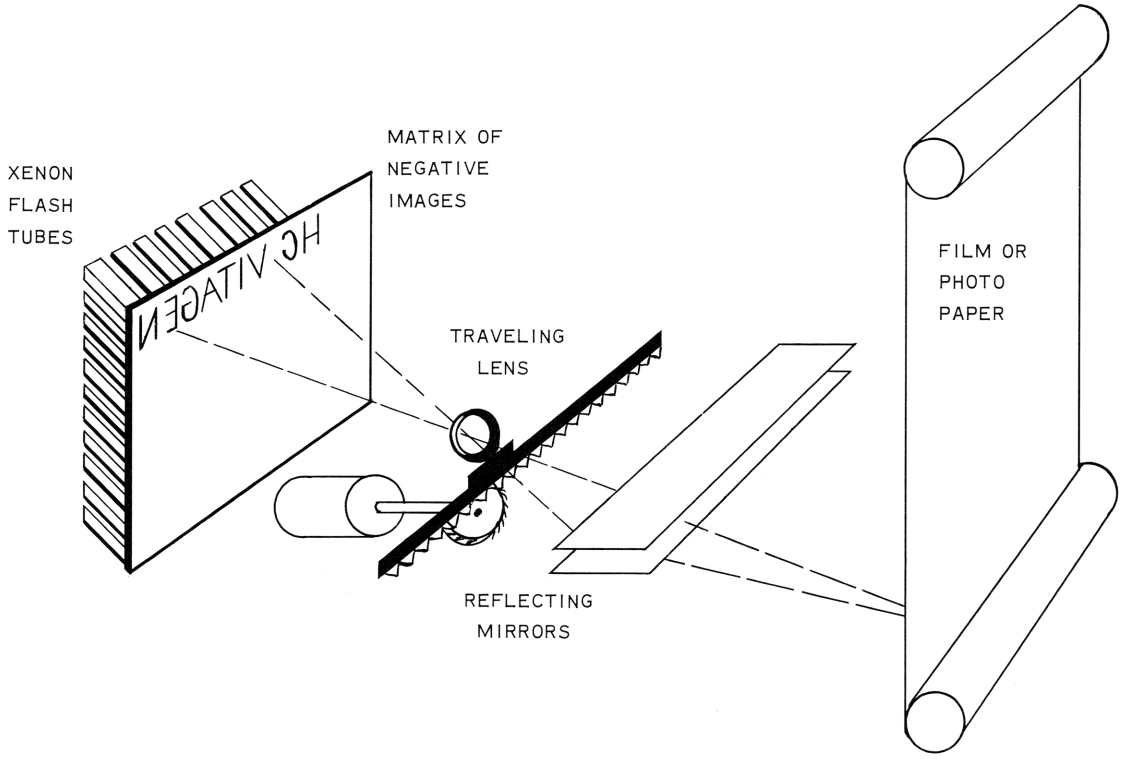
c. is designed to be used primarily to output fully-composed pages of composition, *and*

d. operates at high speeds—in excess of 100 characters per second, and more likely, several hundred or more.

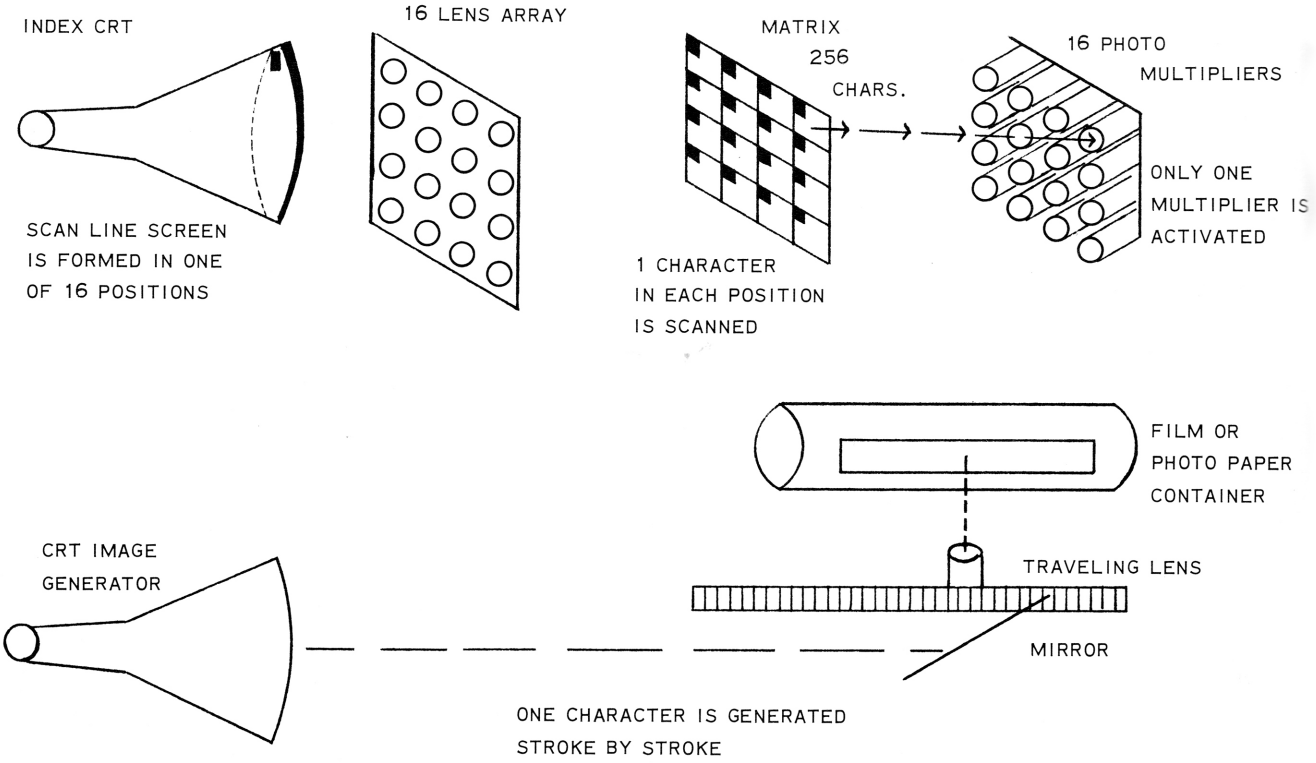
Only one of the above criteria would not be sufficient to distinguish a third from a second generation device. For example, ATF manufactured a "computer slave" typesetter, the ATF-CS, which did not perform "its own" justification function (thus meeting the criterion of point *a* above) but which set type at a speed of perhaps five characters per second, by exposing type images from a revolving (stop-start) disk. The Photon 900 series ("ZIP") operates as a computer slave at speeds well over 100 characters per second, but it generates its characters directly from photographic images. A Linotron 505 meets all of the requirements except that it may or may not have been devised *primarily* to accept input from a computer, since it can do "its own" justification and can accept input directly from paper tape produced from a special keyboard which obtains its width value information by cable connection to the memory store of the photo unit.

Various (second or third generation) devices can be instructed to produce full-page composition, but the facility with which this can be done is one of the features which should distinguish a second generation device from a third. However, the relevance of this criterion is certainly arguable. On the other hand, it is the intimate relationship between the computer and the photo unit that characterizes the third generation device—so much so that we will soon have available to us typesetting machines which also perform many, if not all, of the previously independent computer functions.

Since at this stage in our exposition we have not yet defined or described computer typesetting, our introduction to the third generation typesetter will have to be superficial until we have established a better background out of which to appreciate it. However, we need to cover certain of its features so that we can continue our evolutionary account of typesetting machines, and portray the role of the computer in its developmental setting.




ILLUSTRATING PRINCIPLES OF PHOTON "ZIP" FILMSETTER




ILLUSTRATING PRINCIPLES OF LINOTRON 505

## 1. The Photon 901

Perhaps we might begin by explaining why the Photon 901 (or "ZIP") is not a third generation typesetting machine.

The first Photon 900 (later modified to become the 901) was installed in 1964 at the National Library of Medicine. It was a high-speed machine, being capable of setting complete pages by laying down characters at rates up to 600 per second. It operated completely from magnetic tape that had been processed by computer programs to provide not only justified lines, with all spacing elements accounted for, but also multi-columned pages, with all gutters and vertical spacing elements accounted for. (However, some of these refinements involved software achievements not actually put into effect in the early months of the first installation.) A computer program was also necessary to determine the sequence in which the device laid down its characters across the line.

Characters are stored in the "ZIP" in the form of photographic negatives in a large grid of 11 rows of 24 characters each, with a size range of from six to fourteen points. Different sizes of characters must be stored as such on the grid, allowing, too, for a 4:1 reduction in the image size between negative and typeset representation.

Behind each character there is a "flash" tube. The sequence of exposures is determined by the sequence in which the tubes flash. A traveling lens operates in its shuttle-like fashion, and provides a feedback of its position to a control mechanism which then causes a light to flash when the proper location for that character has been reached, whether the lens is moving from left to right (for one line) or from right to left (for the next). Characters are not laid down in the sequence in which they occur, but in an optimized manner which is determined by computer program. The position of each character across the line is known in advance, and this information has been stored. The only moving element in the photo unit is the traveling lens mechanism, and, of course, the film advance, after each line has been set.

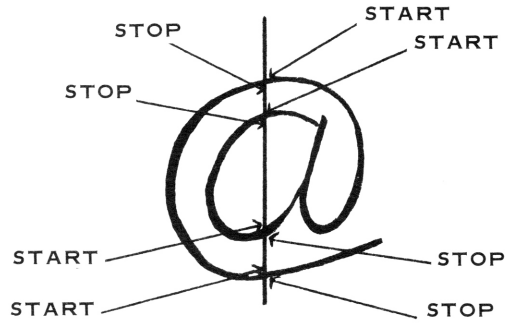
It should be evident that the stored information on each character's position and the proper firing sequence of each flash tube could only be calculated by a computer. Despite the speed of this device we hesitate to classify it as "third generation" because it still uses photographic negatives as a method of producing character images.

## 2. The Linotron 505

The first truly third generation photocomposing device was the Linotron 1010, built by Mergenthaler and the Columbia Broadcasting Laboratories under contract with the United States Government Printing Office. This was followed in time sequence by the Videocomp 820 (marketed by RCA but developed in Germany by Dr.-Ing. Rudolf Hell). However it will be logical to describe the Linotron 505 first since its principles are closer to those of second generation machines.

The Linotron 505 (marketed by Mergenthaler but developed in England by Purdy-McIntosh) also uses a traveling lens system to lay down characters. Moreover, it uses photographic grids as a character storage medium. The interesting difference is that the characters in these grids are not exposed by a light source and projected through a lens system. Instead, the character is identified by a raster and photo multiplier, which reduces the image to a set of signals, each corresponding to a vertical scan of a slice of the character, with on-off signals corresponding to the presence or absence of a portion of the character.

For example:




In the above illustration the slice of the character we have shown involves eight different on-off signals. These signals are transmitted electronically by circuits which cause a similar pattern to be created on the face of a CRT (cathode ray tube). This is done by the deflection of a beam of electrons that bombard the phosphor coating on the inside of the tube. The image so generated is then picked up by a traveling prism and positioned at the proper location along the measure. In other words, in this case it is not the entire character that is laid down along the line by the prism, but only one vertical stroke. The traveling lens moves on, feeds back information on its new location, and asks for another signal from the flying spot scanner. Another vertical slice is then generated. The size of the image on the CRT is controlled by the electronic circuitry so that the same photographic image when scanned can generate different sizes of characters with different proportions.

A resolution of 650 or 1300 vertical strokes to the inch is provided. Different point sizes (really, *set* sizes) will be created, therefore, by fewer or more strokes, generated in the scanning process. Hence the resolution remains the same for all sizes. The vertical height of the character can be elongated or abbreviated by proportionate changes in the stroke, in relation to the original image.

This, then, is the fundamental principle of the 505. Characters are stored photographically on four grids, each capable of holding something less than 256 images (certain blanks are left, to correspond with assigned function codes). The grids rotate into a scanning position by a windmill-like motion. If one grid is in position, and if the characters required are all on that grid, the machine will operate faster than if a different grid must rotate into position, and meanwhile the traveling prism must continue at a constant pace, not being able to wait for the grid shift. When it returns to the same location in the same line (the film not having been advanced under these circumstances) the new grid is in place and ready to be scanned for the next character. Thus a grid shift is a "slow function."

The interposition of a lens element can oblique the strokes in each character, causing the image to take on an italic slant. (In later models electronic obliquing is possible; in either event the stroke on the CRT takes on about a 15 degree angle).

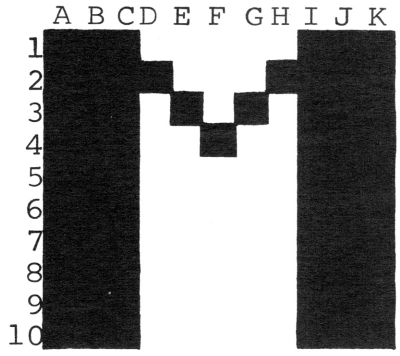
Quite apart from the delays caused by such slow functions as grid shifts, the presence of the moving lens or prism involves the use of a mechanical feature which slows the rate at which characters can be generated. The principal reason for this solution is that characters tend to be distorted on CRT tubes. By using only the very center of the tube, and generating only one stroke, and not an entire character, a truer image can be created, with a less expensive tube, than would be the case if a string of characters were positioned across the entire face of the tube.

### 3. The Videocomp 800 Series

Characters are also generated on the face of a CRT tube by the RCA Videocomp, except that in this case the entire character is created in its proper position within the line, and the image is exposed through a stationary lens system. The only moving part is the film advance after a line has been composed and exposed. A higher quality CRT is necessary for this application, and extensive adjustments have to be made in electronic circuitry to correct for the tube distortion. One model of the Videocomp reduces the image 50% from CRT to film and other models increase image size 2:1.

One point of difference between the Videocomp and the Linotron 505, therefore, is the fact that there is no traveling lens system, and the entire line is generated, stroke by stroke. Each character in that line across the page (whether the format is multi-column or single column) occupies its proper relative position on the CRT. Characters are not all visible at once, although there is some latency to the image. They linger on the face of the CRT only long enough to expose the film or paper, but even if the line could not be seen, the individual characters and, in fact, the strokes comprising the individual characters, march across the face of the tube in their precise position.

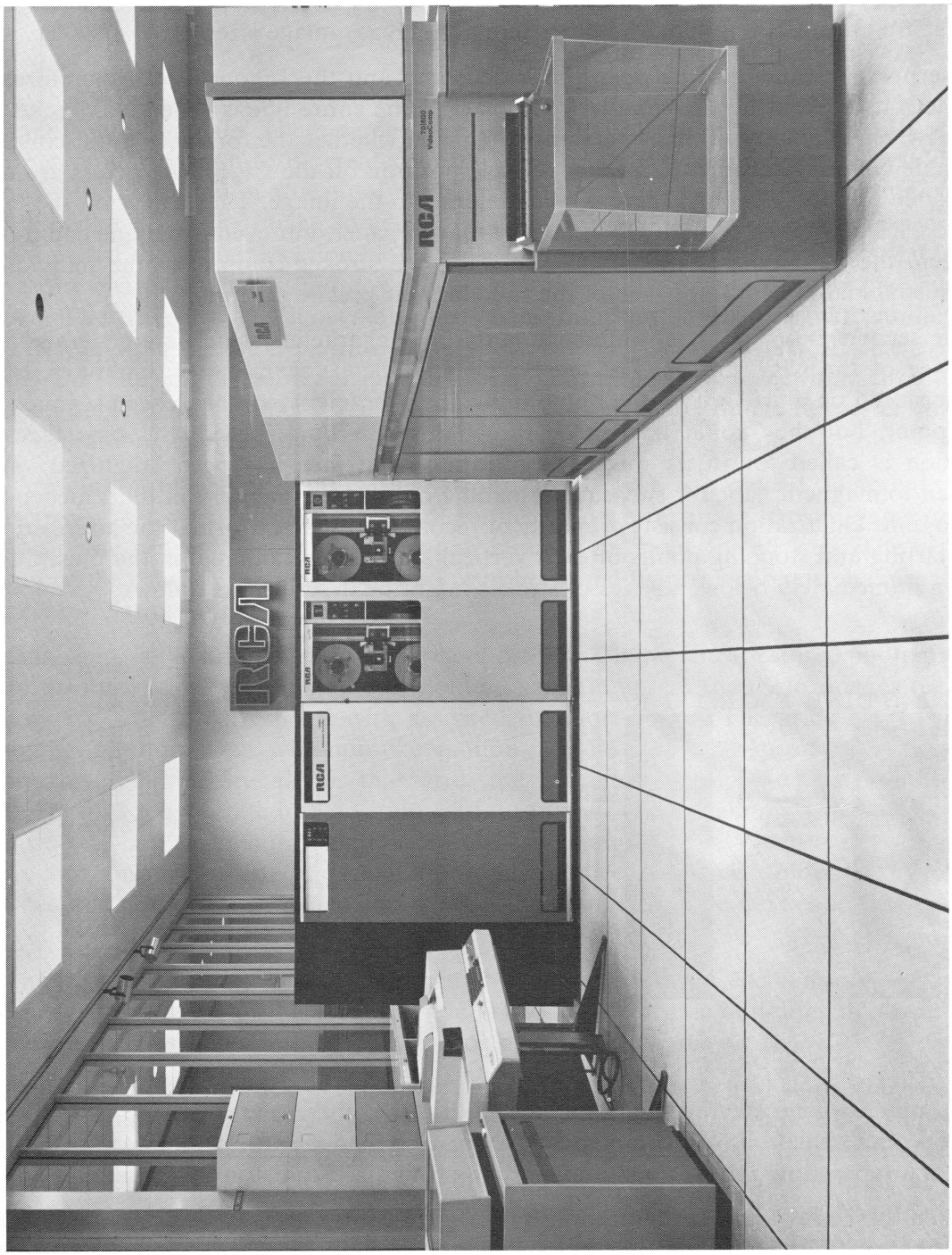
The second major point of difference is that the characters which are generated on the face of the CRT are not created by a simultaneous scanning of a photographic image stored on a grid within the photo unit. The characters may have been scanned at one point, but they could have been drawn instead. Long before the type face in question is called for in its specific application, that face has been "digitized" and written to magnetic tape for subsequent loading onto a disk memory in the Videocomp photo unit. Digitization consists primarily of recording in binary format the addresses of the starting and stopping points of each vertical stroke. For example, in the illustration shown immediately below, a series of strokes might be described as follows:




| *Start* | *Stop* |
|---|---|
| A1 | A10 |
| B1 | B10 |
| C1 | C10 |
| D2 | D3 |
| E3 | E4 |
| F4 | F5 |
| G3 | G4 |
| H2 | H3 |
| I1 | I10 |
| J1 | J10 |
| K1 | K10 |

In order to generate the image on the CRT tube, these codes can be stored in a memory and called out, on command. They can be loaded into memory at the beginning of a typesetting job by maintaining a file of digitized fonts on magnetic tape.

When fonts are stored in digitized fashion it is of course necessary to provide a place to store them. The photo unit has to have access to such storage, either within its own structure, or peripherally, as on a disk or drum. But if they are stored peripherally the access time must be very fast because there is a great deal of information to be transferred.




RCA VIDEOCOMP INSTALLATION, DAYTON, N.J.

Photographic storage of fonts, which are then scanned and digitized on demand, is probably a less expensive way to maintain a ready file of character information, even taking into account the extra equipment necessary to scan and digitize the characters as they are accessed. But there are advantages to digital storage. One is that when characters are scanned and first digitized the results are not always entirely satisfactory. There are problems with digitization which can best be solved by an editing procedure which "cleans up" the character and makes it more presentable. This is especially true where the character has many completely vertical portions, and the flying dot scanner, moving down the vertical edge, may become confused and produce unwanted "on-off" signals. It is also likely to be the case that some fonts digitize more successfully than others, and type designers will wish to make changes in the design of individual characters in order to produce a more effective image, especially when representing diagonals, subtle curves, and delicate serifs. On balance, then, it would appear that pre-digitized fonts should provide better results.

It may be appropriate to observe, while we are on this point, that in the early period of CRT typesetting, before there was much acceptance of this product, questions were raised about the quality of type which could be generated on the face of a CRT from digitized encoding. It is true, of course, that a "step effect" may be discernible. How noticeable this is will be a function of the "resolution" of the character—that is, how many strokes per inch are allowed during the process of character generation. This is a complicated problem, because the more strokes, the more storage area required to describe the character, and therefore the more expensive the device. The more strokes that have to be transmitted, other things being equal, the slower the device will be. Then, too, there is the question of how size changes are accomplished. One method is to digitize each size separately, but this requires too much storage. Another is to increase the size of the dots that make up the vertical strokes, and to space the strokes farther apart, in order to produce larger characters, and to reduce the dot size, spacing the vertical strokes closer together, in order to produce smaller characters. This keeps the same number of strokes per em of type, but it causes a deterioration of character resolution since, as the character becomes larger, there are fewer strokes or lines per inch, and at some point the image would become unacceptable.

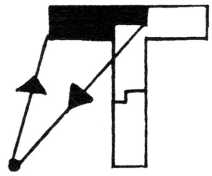
Thus, one of the problems with digitized fonts is the amount of storage required. Even though a great deal of flexibility can be obtained by generating characters from the repertoire in storage at a given time, and by enlarging, condensing or expanding, as well as obliquing them, additional type masters and special characters must also be made available, and when they cannot fit into storage they must be loaded as needed. In the earlier models such loading was accomplished only by adding a relatively long string of digitized information to the text stream whenever the fonts, or subsets thereof, were needed. The computer time required to pull the font information from a disk, write it onto magnetic tape, overlay existing fonts in the photo unit with new ones, and then reverse the process again to restore the original fonts, was a major drawback to the system. Disk storage on line to the photounit was therefore provided for the later models.

### 4. The Fototronic CRT

The Intertype Division of Harris Intertype manufactures the "Fototronic CRT" typesetter, which also uses digitized fonts. It sets a line across the face of the CRT in much the same way as the Videocomp, except that characters are produced on the face of the

The Rocappi installation of the Harris Fototronic CRT. The left cabinet houses the cathode ray tube and film transport. The right cabinet contains the magnetic tape transport, computer elements and storage disk for the digitized type characters.

CRT not by vertical strokes but by dots. Each character is divided into segments. A set of coordinates indicates the location of the first dot in a "patch" of the character. From that dot a pattern is plotted. When the patch is completed the CRT beam returns to its home position and picks up the instruction for painting in the second portion of the character. This process continues until the entire character has been painted. The sequence in which the patches are painted depends upon which patch is picked up first from storage. Since individual dots are stored (two to each byte) a great deal more storage is required for each character than is the case where only start-stop points are stored.



While this method of generating characters on the CRT seems somewhat slower than to direct the electron beam so that it functions merely to produce an on-off pattern of vertical strokes, stepping across the face of the tube by constant increments, it is argued that the inconvenience of returning to the home position and moving out to a new patch is more than offset by the fact that it is not necessary to stroke through those areas of the character where no image exists and the beam is turned off. It is claimed, therefore, that the beam can move more slowly for equivalent speeds, resulting in better positioning and exposure control.

There are fine points to the theory of character generation which would interest only the expert. Whether a character is "stroked" or "painted" the speeds of the Videocomp and the Fototronic CRT for straight setting of text are quite comparable. However, the Fototronic requires a great deal more storage area for its digitized fonts. This would appear to be a substantial liability, but Harris has made a virtue out of it by devising a way of handling the storage of fonts on a magnetic drum or disk with very fast access time, which actually makes a greater range of fonts available, with greater efficacy so far as flexibility is concerned in accessing an extended character repertoire, than is presently true of the Videocomp. The same methods of storage could, however, be used by RCA. One possible liability with respect to the present Fototronic CRT is its inability to store enough data on one track to describe a large enough master to permit adequate resolution in very large sizes of display type. In such cases, Harris will probably have to store stop-start points of vertical or horizontal strokes because there are too many dots in these large masters for convenient storage.

One advantage in storing a character in patches is that the read head on the disk or drum can more quickly start the process of character generation, since it does not have to wait until it finds the beginning of the character.

Another feature of the Fototronic CRT that adds greatly to its utility is its ability to "reverse lead"—that is, to back up the photographic film or paper under careful control, so that multi-column work need not be set across the page. This simplifies the computer processing required in page formatting.

The capabilities of the Fototronic are illustrative of the future promise of CRT typesetting. Not only is negative or reverse leading possible, but a film transport shift permits exposure of a wider page than could otherwise be generated across the face of the tube. Type sizes can be created in half point increments, with the height and width of characters wholly independent of each other. The base line can be altered by deflecting the beam so that inferiors and superiors (subscripts and superscripts) can be represented, thus vastly enhancing the character repertoire. Within a "window" or aperture across the middle of the tube, characters can be set in any location.

Many of these same features exist on the Videocomp, which now also offers a "turn page" feature, permitting pages or portions of pages to be set to longer measures and for vertically-oriented captions or display lines. The newer models of the Videocomp also offer the ability to digitize graphic images from microfilm, and the ability to output typeset microfilm.

Handling of graphics (line work first, half tones later) is moving along very rapidly. Such images can be typeset with excellent resolution. The problems at the moment have to do with techniques for digitizing and storing graphic images, rather than generating them.

### 5. The Linotron 1010

It is doubtful whether the Linotron 1010 will continue to be produced for commercial use. Only one is so employed at this time. The others (two at the U.S. Government Printing Office and three at the Wright-Patterson Air Force Base) are engaged solely in government typesetting applications. But it is nonetheless interesting to describe the features of the machine, especially since it was a pioneer product which would never have come into being except for a government contract to produce it.

Like the 505, characters are generated on the face of the CRT from scanned photographic masters. A four-inch-square glass grid, containing 255 characters and a test pattern, is positioned so that its store of characters can be read. A windmill-like arrangement is available to permit shifting from one grid to any of three alternate grids. Characters are not scanned by a flying spot, but the result is the same. A "cloudlet" of electrons representing the shape of the character is swept over a stationary aperture, causing an analog signal to turn on and off. This signal is then used to generate the character image on the CRT.

The 1010 is unique in that it sets a page at a time, and then advances the film to the next page. This means that the face of the tube must be large enough to contain the entire page (8 ¼ by 10 ½ as enlarged 2:1, from the face of the CRT.) This makes for a very expensive tube, with complex correction circuitry, and also implies an exceptionally high replacement cost.

The positioning of characters on the face of the CRT is therefore controlled not only in a horizontal sense, but in a vertical sense as well. It would theoretically be possible (if the computer software were sophisticated enough) to set characters from one grid, wherever they might be required, and then to shift grids and "spray" additional characters wherever they were supposed to be located—special symbols, display heads and the like.

Character density (resolution) is the same regardless of point size. In this respect, too, the 1010 is like the 505. But unlike the 505, character lengths and widths cannot be controlled independently.[1]

The speed of the 1010 is given as 1280 characters per second for five-point type, 1090 for six, 800 for eight, 620 for ten, 400 for fourteen and 270 for eighteen points. This represents a constant rate at which strokes of the characters are generated on the CRT. In addition, the machine will set twice as fast in a "proofing" mode because it generates only one out of three vertical strokes.

[1] There is a limited capability to expand or condense, but not to specify set widths independently of point sizes.

### 6. The IBM/Alphanumeric 2680

IBM has ceased to market this machine and Alphanumeric has evidently discontinued its manufacture, but a description should be included to provide a complete listing of all such third generation devices—especially so that we may add more input to a discussion of design options for CRT typesetting.

The 2680 creates its characters by vertical strokes in much the same manner as the Videocomp. Characters are first digitized, cleaned up and stored in a special "packed" format which reduces the amount of storage they require. They are loaded onto a disk or drum which runs on-line to a computer. The characters that are required are brought from the disk into computer memory and permitted to reside there during the typesetting application. But if an extended character set is required, or a variety of sizes of type, then a special computer pass is needed to analyze character requirements and incorporate information into the text stream that would trigger the calling of characters from disk into memory, overlaying other characters, so that the desired characters will be in memory at the precise time they are needed, but not in memory at all times. This computer pass is relatively slow and increases the cost of the typesetting operation. The storage of fonts, even in a packed form, is a special problem with this device because each point size has its own digitized master. The 2680 does not enlarge, reduce, expand or oblique type electronically. In a sense, then, its designers were profoundly influenced by "hot type" concepts, or perhaps intimidated by the notion that type would not retain its "true" character if its proportions were changed, or even if the same proportions were maintained in different size ranges. (As a matter of fact, hot-metal type is not completely redesigned for each change in point size, but only as between one range of several point sizes and another range).

One may also dare to suggest that there is nothing inherently sacrosanct about existing type design concepts and standards.

It may be that Alphanumeric's decision to digitize its fonts in this fashion was dictated more by reasons of machine design than aesthetics. It is difficult to see, however, what machine design considerations precluded the electronic manipulation of the digitized character to produce variations on that theme.

There is one other aspect of the 2680 that we should call attention to. That is the method of film advance. We have seen that the Videocomp, the Fototronic CRT and the Linotron 505 set one line at a time. We have seen that the Fototronic CRT can reverse the forward motion of the film. (This is also possible on the Linotron 505.) We have seen that the Linotron 1010 moves film only from page to page. The 2680, as a further distinguishing characteristic, moves its film constantly. Each line of type is set "uphill" to correspond to the steady movement of the film. Thus the time loss, in not being able to set characters while the film is advancing from one line to the next, is saved. (On the other hand, the Fototronic CRT offers "dynamic leading," which sets type uphill only during the interval in which the film is being advanced.)

With the 2680 the coordination of the rate at which characters are formed and the rate of film movement must be taken into consideration. Possibly this dictated the need to design new fonts for each size change, so that there would not be random variations in the rate at which characters are laid down, since character enlargement from given digitized masters implies fewer strokes per inch as point sizes increase. But the concept of a constant film advance, efficient though it appears to be, imposes certain demands upon the computer software, and certain limitations upon the flexibility of the system, which are quite serious.

One final point should be made: all of the other devices are off-line machines. They have to have their own special-purpose computers inside them to handle many functions, but they do not run on-line to a general-purpose computer, as the 2680 does. (An interesting exception is a specially-contrived version of the Linotron 505 at the University of Newcastle-upon-Tyne.) The 2680 needs the computer memory for character storage and for other functions. To run such a device on-line means that it cannot run when the computer is otherwise engaged or that the computer cannot run when it is busy setting type, except as may be possible through multi-programming. To the extent that the computer can operate the 2680 as just another peripheral, this may not be much of a handicap. Still it is another possible limitation in the "system" concept behind the 2680.

### 7. The Photon 7000

A CRT machine made for Photon has now been announced. It will contain most of the features of the Alphanumeric character generation device, plus a mini-computer front end for "driving" circuitry. This machine was developed by Auto-Logic, a subsidiary of Alphanumeric.

### Summary

Our discussion has carried us pretty far afield. But we begin to see how many different factors need to be taken into consideration in thinking of a typesetting system. There are even more, and some of these will be discussed in the next chapter.

### Review Questions

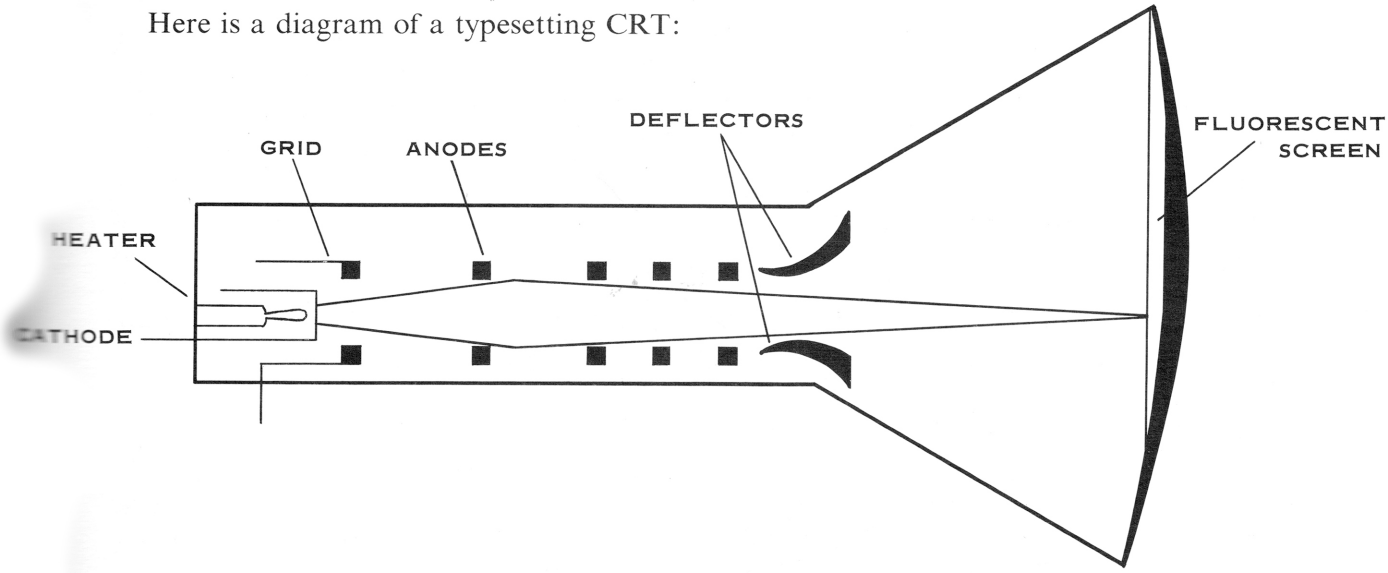
1. How do "third generation" typesetting devices differ from their predecessors?
2. How are character images stored on the Photon "ZIP," the Linotron 505 and the Linotron 1010? How are they stored on the Videocomp, the Alphanumeric 2680 and the Fototronic CRT?
3. What do we mean when we talk about digitized character storage?
4. What is a cathode ray tube? How does it work? (For a further explanation see the appendix to this chapter.)
5. How is justification performed by third generation typesetting devices?
6. What are the merits of photographic character storage as opposed to digitized storage, and vice versa?
7. What is "reverse leading"? What is "dynamic leading"?
8. What are the most significant points of difference between the Linotron 505 and the 1010?
9. How does the Videocomp method of digitizing a character differ from the Fototronic method?
10. Why might it make a difference if a typesetting device is "on line" to a computer rather than "off line"?

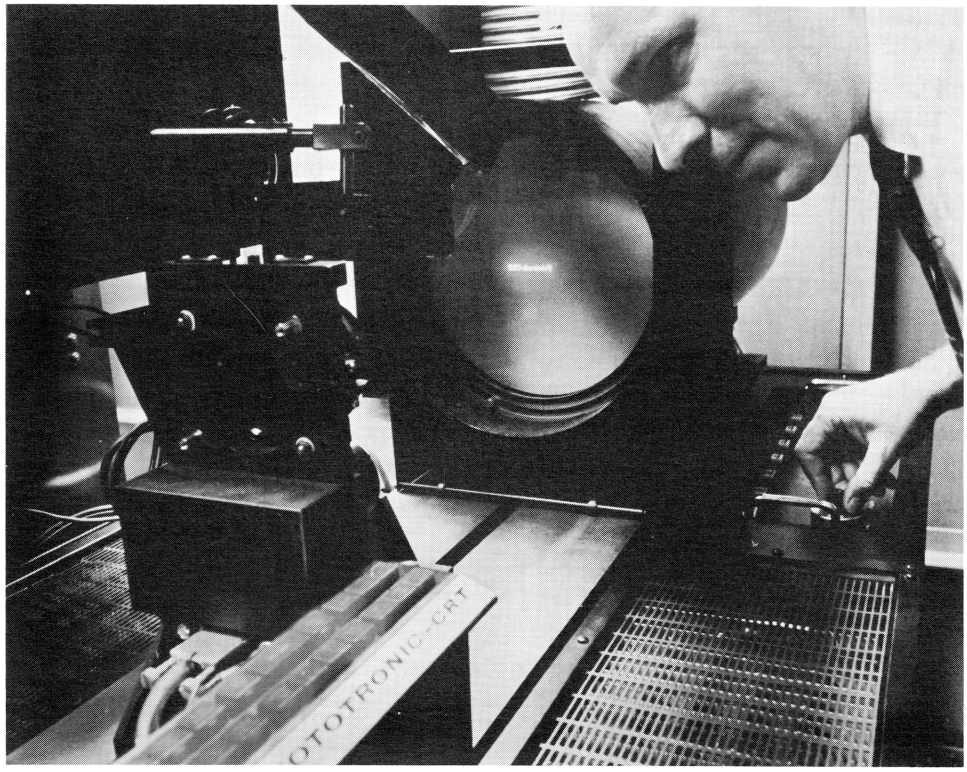
# Appendix to Chapter VI

## How a Cathode Ray Tube Works

This discussion is confined to the CRT in relation to typesetting applications. It is not a general exposition of CRT design or potential.

Here is a diagram of a typesetting CRT:




INSIDE VIEW OF THE PHOTOUNIT ELEMENT OF THE FOTOTRONIC CRT, SHOWING CHARACTERS BEING GENERATED ON THE FACE OF THE TUBE. THE LENS SYSTEM IN THE FOREGROUND DIRECTS THE IMAGE ONTO THE PHOTOGRAPHIC PAPER OR FILM WHICH IS NOT DEPICTED IN THIS ILLUSTRATION.

Electrons are subjected to high voltage acceleration, in an electron "gun". This creates an electron "beam" of charged particles, which is capable of being concentrated or deflected, either by electrostatics or magnetism. A grid pulse controls the emission of electrons. Anodes within the tube focus the beam. Deflection is caused by changes in the potential across horizontal and vertical electrostatic plates or by deflection coils that create right-angled magnetic fields.

There is a phosphor coating on the inside of the face of the tube, and the electron beam, when it strikes this coating, excites the phosphor, causing it to glow. If the phosphor coating is very fine and the beam is closely focused, a very tiny and quite brilliant dot can be created. If the beam is moved vertically, by deflection, then a series of dots, resulting in a vertical line, will be formed. How long the line will be visible on the face of the tube will depend upon the "latency" of the image—the "decay" time of the phosphor. For some applications, where the same image has to remain for a period of time, the image is "refreshed" at, say, 60 frames per second, by recycling the same bombardment of electrons. For typesetting purposes a low degree of latency is desired, but the initial image must be sufficiently brilliant to photograph well.

### Distortion

Since the beam has to travel farther to the edges of the tube, the beam is somewhat more diffused (out of focus) when it arrives. Also, because of the angle at which it strikes, there will be an additional spreading of the dot into an oval shape. Hence, characters produced anywhere except in the center of the tube are subject to some degree of distortion unless there is a correction factor introduced into the electronic circuitry, causing more or less deflection and changing of focus patterns. But applications involving high-quality reproduction generally seek to avoid the use of the outer perimeter of the tube, especially the far corners. The distortion around the corners is called "pincushion" distortion.

Compensating circuitry is also built into the electronics to correct for distortion in all models of CRT devices. The larger the area of the tube used, the more complex the circuitry.

The larger the CRT the larger the "good" center area of the tube and the less the distortion problem. But the larger the tube the more expensive it must be and the more difficult to construct. Columbia Laboratories was involved with Mergenthaler in the design and construction of the Linotron 1010 largely because the full-page feature of that device required an unusually large CRT of exceptional quality, and it was this technology, especially, that the Laboratory could contribute.

CHAPTER VII

# Some Introductory Thoughts About Typesetting "Systems"

Typesetting devices have to be understood and evaluated as part of a "system," with printing as the end product.[1] The viability of the system depends upon the manner in which the components fit together. For example, if the printed product must be produced by letterpress, photocomposition is presently a less logical typesetting solution than hot metal, since in the latter case printing can be done from the type itself (in short-run work) or from plates made from molds taken from the type (in long-run work).

The system must also take into account the work flow—the total organization of production from the time the manuscript is received until the product is produced. The intensive time span for critical production must also be considered. One typesetting system may be good in one application but not in another, depending, for example, upon whether the customer needs to inspect the product at intervals as it is being created, and whether or not he will revise it substantially while it is in an interim stage.

The system also has to take into account the typographical ingredients themselves—the variety of type faces and sizes, the format requirements, the amount of illustrative material, etc. A method which is entirely appropriate and necessary for the production of scholarly treatises in advanced mathematics may be completely inappropriate for setting newspapers or business forms.

In view of the complexity of the tasks to be performed, how can we discuss typesetting systems in an intelligent and reasonably comprehensive manner? Obviously we shall have to approach the topic by a consideration of prototypes or models. Our generalizations will be relevant only to the set of assumptions that the model implies.

[1]The need for additional end products or alternative end products could of course affect the validity of the system.

**The Hot-Metal Shop—Linecasting for Book Work**

Let us assume, then, that we look first at a typical linecasting operation, perhaps engaged in producing books—a relatively specialized application. Here are some of the critical steps in the process.

1. Manuscript is received. Specifications are studied. Copy is marked up, divided according to type sizes and measures required, and assigned to various operators, through their foreman.
2. Material is keyboarded. Slugs are assembled into galleys. All miscellaneous takes of different sizes and measures of type are arranged in sequence by a hand compositor or "bank man." Footnotes are positioned at the end of relevant paragraphs. Proofs are then pulled.
3. The manuscript is re-assembled and the proofreaders check galley proofs against manuscript for the "first read." "Typos" (typographical errors, printers' errors, or "p.e.'s") are marked according to proofreading conventions. Other conditions of the manuscript or galleys may be queried.
4. Marked galleys are returned to the foreman of the machine department so that corrections may be assigned, either to the original keyboarder or to others. These correction lines must be keyboarded in the appropriate type faces, sizes, and measures.
5. Proofs are usually pulled just of the new slugs. The corrected lines are then inserted into position by the bank men and new galley proofs are pulled.
6. The new proofs, the old proofs, and the proofs of the corrected lines only, are returned to the proofroom for checking, together with the manuscript. If some errors still remain the cycle may be repeated. A few errors may be permitted to go uncorrected, though marked, awaiting further revision by the author. Queries raised in the initial reading must be transferred to the new galleys.
7. The manuscript and the new marked-up galleys go back to the publisher, who usually forwards them, unread, to the author.
8. The author reads them carefully; he catches a few undetected typos (the end product will *never* be entirely error-free) and responds to the proofreader's queries. He changes his mind when he sees his material "in print" and makes additional revisions in his text, in the interest of clarity, felicitous expression, or to incorporate later information. He makes these revisions on the galleys or on sheets appended to them.
9. The material goes back to the publisher, who returns it to the typesetter. The marked-up galleys and other supporting documents are sent to the composing room and the necessary lines are set or re-set.
10. These are proofed, then inserted into the galleys, and all proofs go again to the proofroom.
11. After they are checked they again go to the publisher, and usually to the author so that he can see that the corrections and revisions have been made.
12. If there are no further changes, the compositor then proceeds to make up pages. He knows the number of the first folio, and the depth of the desired page. Running heads must now be keyboarded, as well as the folios. The make-up man breaks the galleys into pages, moves footnotes to the bottom of the page, inserts the running heads and folios, spaces out around heads to make pages balance, and moves lines from page to page as necessary to avoid "widows" or illogical page breaks.

13. After pages are made up, page proofs are pulled. These go back to the proofroom to check for balance of pages, spacing, correct running heads, proper front matter, page numbers on the table of contents, etc.
14. Assuming everything is correct, page proofs are sent to the publishers.
15. Uusally the publisher's copy editor or proofreader now reads the full text for the first time since it has been typeset. He may revise the text himself to make the book fit into a desired number of signatures, or he may ask the author to do so. The author gets another look at his product, but is discouraged from making further revisions (except perhaps at his own expense).
16. Revisions nevertheless occur in the made-up pages. They have to do with the aesthetic appearance of the page, the manner in which illustrations are positioned, where heads fall, and how the book divides into signatures. Such revisions can be substantial or minor. Marked-up pages go back to the composing room, through machine composition, make-up, proofreading, back to composition, and finally pages are completed and supplied to the printer, or for negatives for platemaking.

### The System from the Point of View of a Magazine Publisher

In order to visualize procedures from the standpoint of a magazine publisher it will be helpful to reproduce (with the permission of the author, Jerry A. Carlson, managing editor of *The Farm Journal)* a portion of the analysis of current production techniques taken from his unpublished technical report (1970), entitled: *Magazine Writing and Editing on Computer Video Terminals.*

> Tracing a typical article in *Farm Journal* will give a fair indication of the amount of editing that goes into a story before it reaches print. Glenn Lorang, one of *Farm Journal's* field editors, phones in from Spokane, Washington, suggesting a one-page feature on farmers who are starting their own grain-hauling fleet of trucks to bypass the railroad's high shipping rates. The managing editor assigns the story, and for several days the reporter fills notebooks with quotes and facts. He roughs out a manuscript on his office typewriter. The preliminary draft is composed of copy paper taped or glued together, paragraphs circled and marked "insert at A below"; heavy pencil tracks throughout. Without going to the trouble of an actual word count, he can only guess as to its total length. Next morning his secretary retypes the article. Glenn checks it for typographical errors and mails the corrected manuscript to *Farm Journal* headquarters in Philadelphia.. It arrives in the managing editor's office three days later. The final character layout specifies room for 80 lines averaging 52 characters. That is about 70 lines shorter than the original manuscript, so it will need severe cutting to fit. It also needs other editing, so several sections must be rewritten. The rewritten version then goes to a secretary who types a clean copy. This copy, done in five carbons, receives further minor pencil editing and typographical markup. Finally, it is ready to be sent to the printer. Fortunately, the airline pouch service out of Philadelphia is not delayed and the story arrives in the Chicago printing plant the next morning. The foreman of the plant's composing room carries the copy to the Linotype section about 120 feet away. The article then waits its turn at the Linotype work center for about three hours, then the Linotype foreman hands it to a Linotype operator. The operator keyboards the text in galley form.

The lead galley is carried to a proof press, several copies run off, and then the type itself is assigned a rack location to be stored while the galley proof awaits its turn in the proofreading room.

A proofreader matches the galley with original copy and marks typesetting errors on the galley proof. New corrections are set for the galley, which is then sent to the composing-room section. The composing foreman parcels out the correction work to a specialist. This "comp" first inserts the galley corrections in proper position, then begins fitting the type into the page form to match the artist's specifications. It takes about an hour to compose the page. With a final hammering down of type and locking of quoins, he shouts for a proof boy to run several white proofs at the nearby hand press. The proofboy is somewhere else, so the comp, with appropriate descriptive phrases, runs off the page proofs by himself.

Later, the comp foreman hand-carries this proof and others 200 feet to the proofreading room. The proofreader marks all the errors she finds on the proof, then copies by hand all the changes again on a duplicate proof, since one copy will be going out to the customer and another retained at the plant as a back-up.

Now signed proofs and increasingly smudged, dog-eared copy is sent back to the composing-room foreman, who removes one signed proof, puts it in the air express pouch for delivery to the authors, and piles the remaining material on a shelf. The pouch containing the proof will be 800 miles away in Philadelphia next morning, if the messenger service works well between airports on both ends, if the planes are on schedule, and if the airline people do not bury the pouch in an unknown storage igloo under tons of other air freight.

The editor, busy getting three other stories into shape for sending to the printer, sees correcting the proof as an unpleasant chore which can be delayed until the last minute. That time arrives quickly, and the editor's pencil comes down on the carefully typeset and corrected proof. Six lines too long! It had seemed like butchery to cut the original manuscript down to 80 lines. But he deletes a couple of words in each of three paragraphs, pulling up three hangers and shortening the article by three lines. One sentence that is not too essential is in the top of a long paragraph. Cutting it would mean resetting the entire paragraph. A more essential sentence near the end of a paragraph could be dropped without very much expensive resetting. He elects for editorial integrity and deletes the expensive-to-cut sentence. That is two more lines out. Here is another possible hanger to cut: If he deletes these two words, will that two-word hanger come up to the line above? He measures with hash marks on a slip of paper as a ruler. Maybe it will fit, if the lino sets it tight.

Again, the pouch courier service willing, the proofs travel between Philadelphia and Chicago. The composing-room foreman checks over the editor's proofreading corrections, watching for any proofreader's queries the editor may have missed. He brackets in red ink the lines that will have to be reset—in this case, about 10 percent of the story—and carries the proof to the lino section. Corrected slugs again await a composing specialist, who inserts them and cleans up the page for proofing. The proofreader again goes over this hopefully final proof—now on yellow paper instead of white—and if all is well, the page will be ready to send to the foundry for press plate making.

By now, the final yellow "foundry" proof has arrived in Philadelphia. Somehow, this yellow color stimulates an editorial scrutiny that the white proof could not. It is a last chance to fix an oversight before the presses roll. In this case there were no errors or changes of heart after the plate was made. Many other articles are not so clean and fortunate, which means a second plate must be made—a "Number Two" that usually costs the publisher roughly twice as much because of foundry overtime charges.

Perhaps one proof out of five also goes through a "revise" stage, where the corrections are so heavy an editor cannot count lines and achieve a fitted page the first time. In a big, efficient job shop where every customer waits his turn, that can add a day of time before foundry plates are made. A substantial share of an editor's time—perhaps 30 to 50 percent—will be spent with such mechanical copy fitting and related processing chores.

### The Author-Publisher-Typesetter Relationship

It may be seen, then, that a "system" approach to typesetting cannot exclude from consideration the relationship between the typesetter and the publisher, or the relationship between the publisher and the author.

"Under the present technology," writes Leonard Shatzkin, in Publishers' Weekly,[2] "—and ever since Gutenberg—the only way to be sure of what was happening in composition was to check the results at each step. The naive could talk about the desirability of a perfect manuscript. But the initiated knew that even with a perfect manuscript the compositor could easily strike the wrong key. If he struck the right key, that channel of the Linotype machine could contain the wrong matrix, or if it had the right matrix, it could be raced out of position by another matrix as it dropped toward the assembling elevator. One operator might set loose lines with excessive space between words, and the man next to him, who did half the book, or set the corrections, might set too tightly.

"Then there were—I should really say there *are*—the hazards of type metal too hot, so that it is too weak to print well or mold well, or too cold, so that the fine serifs are lost and the surface is not smooth. Perfectly good slugs can be dropped, or transposed or lost. If more than one man is working on page makeup, widows in the front of the book may be dealt with differently from widows in the back of the book. In short—if you care about quality—there is really no way to escape the need to verify at *every step of the process.* Hence the practice of supplying the customer with galley proof, page proof, revised page proof, stone proof, repro proof and blueprints.

"With what result? How often have you seen a book completely free of error? Well, one result is this. Proof is unavoidably necessary to catch errors in transcribing the manuscript, but it provides the opportunity for reconsidering the manuscript itself, in type. We all know how low is the resistance to that opportunity. That is human nature and virtually uncontrollable so long as the opportunity exists. As a matter of fact, we are so used to the idea of doing this—which we call author's alterations—that we imagine that the proofs are submitted for the purpose of author's alterations, rather than catching printer's errors, which is, of course, complete nonsense.

"While the printer pays for correcting his own mistakes, the publisher pays for these author's alterations. How expensive are they? Well, at McGraw-Hill, they ran, for the one year I totalled them, $456,000. We can certainly put that money to much better use."

[2] "Some Effects of Computer Composition On the Editorial Process in Book Production," Publishers' Weekly, April 7, 1969, p. 28 ff.

Shatzkin goes on to analyze the nature of author's alterations and finds that, in general, they divide themselves into two categories:

a.) Changes that could easily have been anticipated in the manuscript and did not require typesetting to become apparent.

b.) Changes that simply represented two points of view, with the second no better than the first. The copy-editor or the author hyphenated—the proofreader unhyphenated—or vice versa.

> So the first cost of verification by seeing proof is *money*. The second cost is *time*. The cost in time is greater than you might imagine, because it extends far beyond the time required to make the changes.
>
> Predicting the precise time that proof will be returned is impossible, and even if it were possible, the amount of work to be done varies greatly and can only be estimated by looking at the proof itself. That is why no intelligent compositor attempts to schedule composition corrections in advance. In place of scheduling, he (and the publisher for that matter) resort to queuing. When the work comes in, it takes its place in line—and the more work in line, the better the assurance of an even flow and lower cost. That is where the book spends most of its seven manufacturing months—on a series of waiting lines at the publisher, the author and the printer.
>
> Actual production time in a manufacturing schedule of seven months is probably less than one month, which may indicate how costly in time is our verification through proof.

How can this arrangement be changed? Shatzkin concludes by stating:

> It is the possibility of moving book manufacturing from the handicraft stage closer to the production line—from the whittling by designer, production manager, compositor, printer, binder, passing the willow stick back and forth among the group, each taking his turn—to an automated production line with controlled and predictable results—it is in this possibility that computerized composition offers its greatest hope.

In terms of the relationships between author, publisher and compositor, we can visualize a complete restructuring and reassignment of responsibilities. We shall appraise them as we go on. We shall find that one concept implies not passing on the manuscript to the typesetting until it is perfect and all the specifications have been determined, and even coding for automatic processing. A second alternative would perhaps afford even more opportunity for author and publisher intervention than now exists, by on-line, interactive computer programs and systems.

But in either case there will be a change, and it is necessary not to lose sight of the implications of this fact as we proceed, because it means opening a new kind of dialog with all of those who are involved in the process, and they must all become familiar with the opportunities and the choices.

**Review Questions**

1. For the kind of typesetting you are most familiar with, can you make up a flow chart of the "system" in present use? Include the author and the publisher in this process.
2. Do you really know what goes on at the publisher's, before you get the manuscript? Could you offer to the publisher alternative ways of helping him to perform some of his tasks? Do you think it is relevant to restudy the relationship between the typesetting and the publisher? Do you think that there would be a sympathetic response to this suggestion?
3. Where would you look for change to come? From the publisher, demanding it? From the supplier, suggesting it? From the manufacturer of equipment, laying out guidelines for new processes?

CHAPTER VIII

# Basic Computer Concepts

Before we begin to consider in some depth the application of computers to typesetting it will be necessary to understand something about computers and computer programs. This chapter will deal with the design and function of computers and the following chapter will be concerned with computer programs and programming.

## The Structure of a Computer

A computer usually consists of three components:

1. one or more input devices,
2. a main frame or central processing unit (CPU),
3. one or more output devices.

Later on we shall discuss input and output at greater length. For the moment let us merely suggest that input can come from (among other sources) punched cards, punched paper tape, magnetic tape, or from scanning and sensing devices.

Output can be directed toward a printer (usually called an on-line printer), or a console typewriter. Punched cards and magnetic or paper tape can also serve as output. Output sometimes consists of signals that activate some other mechanism.

At this point we wish to direct our attention to the main frame or CPU, which is connected to input and output devices by means of *channels.* Channels are also used to connect other peripheral devices which serve as a temporary store of data, such as magnetic tape drives or disks.

Input data flow from the input device to the CPU over channels, and from the main frame to the output device over similar or identical channels. Channels also access interim storage peripherals, and these devices provide an intermediate resting place for input or output, although they are also used for reference purposes and as depositories for partially processed data.

A channel not only *transmits* the flow of data, but also regulates and controls this flow so that it is synchronized with other operations. When a peripheral input device is ready to transmit a batch of information, or when the CPU is ready to accept it, the channel controller will be signaled. It will then accept the data or a manageable slice of it, often in a fashion which permits the virtually simultaneous transmission of other data in an overlapping fashion.

There are often two kinds of channels: multiplexor channels and selector channels. A multiplexor is used for slow peripheral equipment and breaks the channel down into what could be thought of as a number of sub-channels. Actually, data are transmitted in

minute particles in some order or sequence, from different devices, and are assigned work or storage areas which in effect permit the assembling and processing of data from one unique source, relating to one unique task, without regard to the fact that other information has been intermingled during the course of transmission. The parallel example in the telephone industry is the ability to transmit many different conversations over the same wire at the same time without any user being bothered by, or even aware of, the fact that others are sharing the same line.

A selector channel, on the other hand, generally conveys information from only one source at a time, in what is called a "burst" mode—which is a very rapid infusion of data which fully occupies the capacity of that channel.

To make it possible for channels to be used in these two ways, and especially to be capable of accepting several different kinds of assignments, such as to transmit data from input device to CPU on one occasion and from CPU to output device on another, there are controllers which operate in conjunction with the CPU to accept and process data in multiplexor and burst modes. Controllers vastly enhance the efficiency of the main frame, since the latter can be commanded to work on tasks already in hand rather than to wait until new data are transmitted. Similarly, the processing of new information need not be delayed until the old has been disposed of (as output) since the CPU can continue to perform available work. In other words, the main frame is not so likely to be "input-bound" or "output-bound," and it can spend its time doing computations, rather than having to wait for input or output devices to perform their tasks.

Channels and their controllers also provide the main frame with certain other services: they will interrupt an operation to inform the CPU that an error condition exists, or that the input has been terminated. They will also perform certain "storage protect" functions which prohibit the inadvertent destruction of stored information by preventing data from being written on top of useful tables, program areas, etc. "Read only" features exist which permit data to be picked up from a given location, but nothing new may be written to that location.

### The Main Frame or CPU

The main frame generally has these elements:

1. a control unit,
2. a storage unit,
3. an arithmetic and logic unit,
4. a set of registers.

The control unit interfaces with the channels and their controllers, and it also coordinates the various functions within the main frame itself. Control is partly "hardware" (wired instructions built into the computer), partly "software" (programs), and it can also be partly "firmware" (i.e., programmed instructions which change the logic paths of the machine as they might otherwise be hardwired). It will not be necessary to our present purposes to discuss the intricacies of control. But it should be obvious that the ability to execute commands and keep track of data is essential to the functioning of the computer.

### Storage

Data coming into the main frame needs to be put somewhere until it is processed, tucked away again in the interim stages of processing, and then arranged in a store

somewhere until it can be presented as output. Data (including the programs themselves) will be pigeon-holed into "locations" which have "addresses" so that the information can be found when it is needed.

Data elements are stored in *core* (or on disk, drum or tape) in the form of "bits," "bytes," and "words." A bit is the presence or absence of an event which can generate an electric pulse--whether the event is a hole in a paper tape or a positively or negatively charged minuscule doughnut of ferrite. A bit is a switch, in a sense, which is in an *on* or *off* position, and can be interrogated in order to find out which position governs at the moment. Since we can therefore consider only two conditions we can describe them as either "0" or "1"—"no" or "yes." Two bits taken together can describe four conditions:

00
01
10
11

In like manner, three bits can describe eight:

000
001
010
011
100
101
110
111

Five bits can describe 32, six bits can describe 64, seven bits can describe 128, and eight bits can describe 256 different conditions.

Six-, seven- or eight-bit "bytes" are used most frequently, especially in typesetting. But the way in which characters are stored in memory is a function of the size of the "word," or string of bits that are hitched together at one address and are generally manipulated at one time (moved from one place to the other, for example). In the IBM 360 series computer, four bytes make up one word, and two bytes are considered as a "half word."

For purposes of doing arithmetic, a four-bit byte may be sufficient. If we wish to deal only with decimal numbers we have need only for 0 through 9, plus a few control characters which should stand for functions like "end of message," or "end of record." But for the treatment of alphanumeric data, six or more bits in a byte are more convenient. If we are not concerned with typographic detail (such as upper and lower case characters) and if our character set or repertoire can be limited, a six-bit byte will be more than ample, because we would only need:

26 letters
10 numbers
10 punctuation marks (approximately), and
6 or so control codes
52

Therefore all of the desired characters can be described with six bits, and a computer printout device (such as a high-speed on-line printer) with a print chain of 64 characters is quite adequate.

For alphanumeric data processing, then, we can visualize computer storage as consisting of a core or memory which looks something like this:

| *Location* | *Bit Configuration* |
|---|---|
| 0 | 0 0 0 1 1 0 |
| 1 | 0 0 0 1 0 0 |
| 2 | 0 0 0 1 0 1 |
| 3 | 0 0 0 1 1 1 |

This could stand for the storage of the number 6457, beginning at location "0."

If we wanted to store the word "word" beginning in location "3," that portion of core might look something like this (in this instance we are using an arbitrary code structure known as TTS to describe the letters we wish to store):

| *Location* | *Bit Configuration* | |
|---|---|---|
| 0 | 0 1 1 0 0 1 | (W) |
| 1 | 0 0 0 0 1 1 | (O) |
| 2 | 0 0 1 0 1 0 | (R) |
| 3 | 0 1 0 0 1 0 | (D) |

### The Concept of Parity

Usually, for purposes of assuring machine and transmission accuracy, an additional bit is provided for in core, and it may also be present on the paper tape or magnetic tape which transmits information to and from core. The additional bit serves as a check to see if the device which writes out or reads the codes is functioning properly. This is a "parity check." Wiring and logic are added to the computer to cause it to count the number of "1" bits in a byte. If this number is odd, a seventh "1" bit will be added to make it even (or vice versa if the parity check is supposed to be odd).

| *Parity bit* | *Other bits in byte* | *Without parity* |
|---|---|---|
| 1 | 0 0 0 0 0 1 | odd |
| 0 | 1 0 0 0 0 1 | even |
| 1 | 1 1 0 0 1 0 | odd |
| 0 | 1 1 0 0 1 1 | even |

A parity check means that the hardware must therefore always be roomy enough to accommodate an additional bit for purposes of error checking, even though the parity bit is ignored for all other purposes.

### What is a Convenient Byte Size?

In a later chapter we will discuss special problems of typographic representation, storage, and information transfer, which arise from our typesetting requirements. At this point we need say no more about the manner in which characters are stored and described other than to indicate that the trend is toward the use of eight-bit bytes, providing 256 distinct code combinations. The code structure which eight bits affords is called "hexadecimal," by which is meant that the numbering system most convenient to this code structure is not based on a "set" of ten, but rather on a "set" of sixteen. (A numerical "set" is based on the number of *single digits* used for counting purposes.) "Sets" are "new math" concepts well understood by children but not so easily grasped by adults who learned their arithmetic some time ago. Perhaps we should digress briefly to talk about sets, and particularly the hexadecimal set. But first let us look at binary and octal sets.

A binary set would be made up of only two numbers, zero and one, as follows:

| *Binary* | *Decimal* |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 10 | 2 |
| 11 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |

It may be observed that in a binary set:

$$1 = 2^0 \text{ or } 1$$
$$10 = 2^1 \text{ or } 2$$
$$100 = 2^2 \text{ or } 4$$
$$1000 = 2^3 \text{ or } 8$$
$$10000 = 2^4 \text{ or } 16$$
$$100000 = 2^5 \text{ or } 32$$

The binary set is made up of only two numbers, 0 and 1, but it can be used to express decimal concepts. The table we have shown above, setting forth binary and decimal numbers, illustrates this. On the other hand, an "octal" set would consist of only eight numbers, from 0 through seven. Here is the relationship between octal and decimal:

| *Octal* | *Decimal* |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 10 | 8 |
| 11 | 9 |
| 12 | 10 |
| 13 | 11 |
| 14 | 12 |
| 15 | 13 |
| 16 | 14 |
| 17 | 15 |
| 20 | 16 |
| 21 | 17 |
| 22 | 18 |
| 23 | 19 |

But an octal set can be described in binary numbers as well as in the arabic numbers 0 through 7:

| *Binary codes* | *Octal* | *Decimal* |
|---|---|---|
| 000 000 | 0 | 0 |
| 000 001 | 1 | 1 |
| 000 010 | 2 | 2 |
| 000 011 | 3 | 3 |
| 000 100 | 4 | 4 |
| 000 101 | 5 | 5 |
| 000 110 | 6 | 6 |
| 000 111 | 7 | 7 |
| 001 000 | 10 | 8 |
| 001 001 | 11 | 9 |
| 001 010 | 12 | 10 |

Notice that if the six bits are divided into two parts (and we have so represented them for convenience) it is easy to describe the configuration of bits in octal terms. Thus:

000 000 is an octal 00
000 100 is an octal 04
001 001 is an octal 11
010 100 is an octal 24

This is a very convenient fact, since we can then create a "table look-up" arrangement which looks like this:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 8 | 9 | A | B | C | D | E | F |
| 2 | G | H | I | J | K | L | M | N |
| 3 | O | P | Q | R | S | T | U | V |
| 4 | W | X | Y | Z | ! | @ | # | $ |
| 5 | % | ¢ | & | * | ( | ) | - | = |
| 6 | ; | : | ' | " | , | . | ? | / |
| 7 | | | | | | | | |

The vertical row of numbers outside the block represents the first digit of a 2 digit octal number and the horizontal row on top of the block represents the second digit. Thus an octal 05 is a "5" and an octal 15 is a "D."

The characters shown in the above grid are assigned arbitrarily, and not all of the codes have been used up, but it will be noted that we have made a decimal 8 an octal ten; a decimal 9 is an octal eleven, and the letter "A" is an octal twelve. If we had an octal 21 (namely 010001) we would know it was an *H,* and if we had the code combination 100000 our table would tell us it was an octal 40, and hence a *W.*

**The Hexadecimal System**

The same principle is involved in the hexadecimal system, except that our numbers set, instead of consisting of eight digits, 0 through 7, consists of 16, 0 through F. In other words, the set counts out as 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

| *Decimal* | *Hexadecimal* |
|---|---|
| 9 | 9 |
| 10 | A |
| 11 | B |
| 12 | C |
| 13 | D |
| 14 | E |
| 15 | F |
| 16 | 10 |
| 17 | 11 |

And so on, until we reach:

| | |
|---|---|
| 25 | 19 |
| 26 | 1A |
| 27 | 1B |
| 28 | 1C |
| 29 | 1D |
| 30 | 1E |

Then 31 becomes "1F" and 32 is hexadecimal "20."

When this theory is understood we can then discuss "binary coded hexadecimal," which looks like this:

| | |
|---|---|
| 0000 | is a hexadecimal 0 |
| 1000 | is a hexadecimal 8 |
| 1001 | is a hexadecimal 9 |
| 1010 | is a hexadecimal A (which is a number) |
| 1011 | is a hexadecimal B |
| 1100 | is a C |
| 1101 | is a D |
| 1110 | is an E |
| 1111 | is an F |

Just as three bits can enable us to count from 0 through 7, so four bits can count from 0 through 15, and the number 16 (in decimal set) brings us to the second or left-hand group of four bits:

| | |
|---|---|
| 0001 0000 | is a hexadecimal 10, although a decimal 16 |
| 0001 0001 | is a hexadecimal 11 |
| 0001 0110 | is a hexadecimal 16 |
| 0001 1111 | is a hexadecimal 1F (decimal 31) |
| 0010 0000 | is a hexadecimal 20 (decimal 32) |
| 0010 1111 | is a hexadecimal 2F (decimal 47) |
| 0011 0000 | is a hexadecimal 30 (decimal 48) |
| 1111 1111 | is a hexadecimal FF (decimal 255) |

A translate table can be prepared which shows the two hexadecimal characters as coordinates. In this case the left-most digit is usually shown at the top, and the right-most digit at the side:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 1 | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| 2 | W | X | Y | Z | a | b | c | d | e | f | g | h | i | j | k | l |
| 3 | m | n | o | p | q | r | s | t | u | v | w | x | y | z | ! | @ |
| 4 | # | $ | % | ¢ | & | * | ( | ) | - | = | ; | : | ' | " | , | . |
| 5 | ? | / | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | | | | | | | | | | | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | | | | |
| E | | | | | | | | | | | | | | | | |
| F | | | | | | | | | | | | | | | | |

Here we have used both upper and lower case letters, arabic numbers, and all the punctuation marks on our typewriter, and we still have not begun to fill up the matrix of characters.

The characters we have shown in the above matrix are again arbitrarily assigned and this assignment does not accord with any code structure in common usage. The reader may be interested to study two different code structures which are commonly employed for information transfer. One is called EBCDIC (pronounced "Ebsadick") which stands for "Extended Binary Coded Decimal Interchange Code," and the other is USASCII ("Yousaskey") or ASCII ("Askey") which stands for United States of America Standard Code for Information Interchange. We have chosen to show both EBCDIC and ASCII hexadecimal codes on one diagram. Showing them this way does not emphasize sufficiently the number of blank spaces (although some of them are taken up with information-handling codes rather than information-content codes). However, it will be useful to note the lack of correspondence between these two code structures. Alarming as this may seem, it is a simple matter to translate from one structure to the other so long as the same characters appear on both grids.

Left half of byte in hexadecimal

| Right half of byte in hexadecimal | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0 | 0000 | | | | | Sp<br>Sp | &<br>0 | -<br>- | | | | —<br>@ | P | | | \<br>\ | o<br>p |
| 1 | 0001 | | | | | !<br>! | 1<br>1 | / | | a | j | A | Q | Ā | J | a | 1<br>q |
| 2 | 0010 | | | | | "<br>" | 2 | | | b | k | s<br>B | R | B | K | S<br>b | 2<br>r |
| 3 | 0011 | | | | | #<br># | 3 | | | c | l | t<br>C | S | C | L | T<br>c | 3<br>s |
| 4 | 0100 | | | | | $<br>$ | 4 | | | d | m | u<br>D | T | D | M | U<br>d | 4<br>t |
| 5 | 0101 | | | | | %<br>% | 5 | | | e | n | v<br>E | U | E | N | V<br>e | 5<br>u |
| 6 | 0110 | | | | | &<br>& | 6 | | | f | o | w<br>F | V | F | O | W<br>f | 6<br>v |
| 7 | 0111 | | | | | '<br>' | 7 | | | g | p | x<br>G | W | G | P | X<br>g | 7<br>w |
| 8 | 1000 | | | | | (<br>( | 8 | | | h | q | y<br>H | X | H | Q | Y<br>h | 8<br>x |
| 9 | 1001 | | | | | )<br>) | 9 | | | i | r | z<br>I | Y | I | R | Z<br>i | 9<br>y |
| A | 1010 | | | | | ¢<br>* | !<br>: | ^<br>^ | :<br>: | | | J | Z | | | j | z |
| B | 1011 | | | | | .<br>+ | $<br>; | ,<br>, | #<br>, | | | K | [<br>[ | | | k | {<br>{ |
| C | 1100 | | | | | <<br>, | *<br>< | %<br>% | @<br>@ | | | L | \<br>\ | | | l | \|<br>\| |
| D | 1101 | | | | | (<br>- | )<br>= | _<br>_ | '<br>' | | | M | ]<br>] | | | m | }<br>} |
| E | 1110 | | | | | +<br>. | ;<br>> | ><br>> | =<br>= | | | N | ^<br>^ | | | n | ~<br>~ |
| F | 1111 | | | | | VR<br>/ | ¬<br>? | ?<br>? | "<br>" | | | O | -<br>- | | | o | DEL |

The EBCDIC code occupies the top of the box. The USASCII eight-level variant occupies the bottom of the box.

**The Instruction Set**

We have established the fact that a computer can perform certain input-output operations (generally called I/0), utilizing its channels and its controllers. It can also store data in core or memory. But how does it manipulate the data? How does it "compute"? These are functions which are performed by the use of its arithmetic and logic unit, with the assistance of its registers, storage capabilities and control unit. What *are* the arithmetic and logical functions it can perform, with the assistance of its instruction set?

Different makes and models of computers have different instruction sets depending upon their design characteristics and the applications for which they are intended. An instruction is called an operation, and it is expressed by an "op code." The command tells the computer what to do and where to do it, or, where to get the data and, after performing the operation, where to put it. Operations may be *declarative,* which defines the elements to be worked with, and *imperative,* which then orders something to be done with them. In the latter category there may be imperative commands of an arithmetic nature (add, subtract, multiply or divide), or of a logical nature (do this if this *and* this are true; do this if this *or* this are true; do this if you have reached this tally) or of a manipulative nature (move this left, or right, fill with zeros, pack or unpack the data, translate it, etc.). Here are a set of imperative commands, simplified:

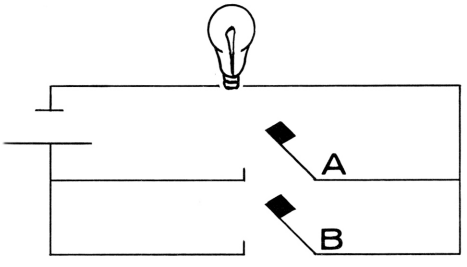
Add
AND
Branch
Compare
Convert
Divide
Exclusive OR
Execute
Load
Move
Multiply
OR
Pack
Read
Shift
Start
Store
Subtract
Test
Translate
Unpack
Write

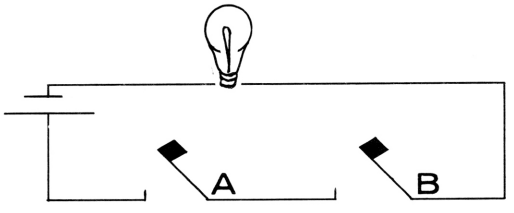
These instructions (and others more specific) are effected by specified commands contained in a "program" but they are executed by the arithmetic and logic unit. The wiring of the unit can permit it to perform certain tasks very fast. For example, the principle behind binary addition is that

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

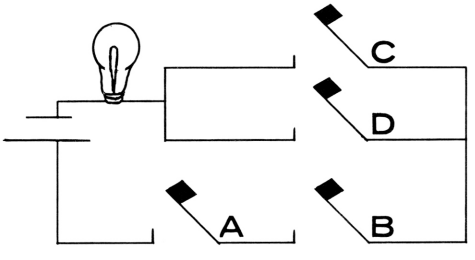
Instructions are executed electronically by invoking the use of certain circuits which are able to perform logical functions. An illustration of how this may be achieved may be shown very simply.

Two circuits in parallel may be described as A OR B, which is the same as A + B. The plus sign is the equivalent of two switches in parallel, as may be seen from the following diagram. If both circuits are open (0 + 0) the result is no current (0). If either is closed (0 + 1 or 1 + 0) there is current and the result is 1. If both are closed the result is still 1. (A shift to 10 is implied.) Multiplication can be performed by a series circuit. If both switches are open ($0 \times 0 = 0$) or if only one is open ($0 \times 1 = 0$), the results will correspond with the basic principles of multiplication, and this is equally true where both switches are closed ($1 \times 1 = 1$).




A more elaborate proposition can be stated as: if A and B are true and either C or D, then make an affirmative decision (light the bulb). The following diagram illustrates this logical possibility.




Obviously these are very simple examples, but they show that electronic circuitry can perform logical operations. If the circuitry consists of diodes which permit current flow only in one direction in a circuit, and inhibit current flow in the opposite direction, very complex conditions can be postulated.

### The Use of Registers

Registers are devices for storing one or more words in order to facilitate arithmetic, logical or manipulative operations, such as the address of the next instruction, a counter of the number of operations performed, a check to ascertain the existence or non-existence of a certain condition, or a quantity or constant to be used in making certain calculations. Data movement operations from one register to another may be much faster than from core to core or from core to register. In some computers all movement of data is accomplished through registers. In others there may be a direct transfer of information from one core location to another.

### Summary

Data are brought in for processing by various input devices. Programs must also be fed into the computer to direct a certain sequence of logical operations. How programs work will be the topic of the next chapter. It is the CPU or main frame that performs the logical and arithmetic operations, utilizing peripheral input and output devices and external storage media as well as core memory so that the various commands can be executed. Channels and controllers help keep track of traffic and try to assure the efficient use of the "compute" power of the main frame. This power is determined by the size and speed of the computer and also the variety of instructions it has been wired to execute. In later models of computers some instructions can be developed by circuits that are built by programs rather than hard wiring, or combinations of wiring and programmed instructions may add to the power of the computer to implement a wider range of commands.

### Review Questions

1. What is the main frame or CPU?
2. What elements does it contain?
3. What do we mean by "peripheral" devices?
4. What is I/0?
5. What is the difference between octal and hexadecimal?
6. What functions do channels perform?
7. What do we mean by the "instruction repertoire" of a computer?
8. What is the difference between a selector channel and a multiplexor channel?
9. Define bit, byte and word.

CHAPTER IX

# Programs and Programming

Programming is a very complex subject. All that we can hope to accomplish is to give the reader a feel for the processes and procedures involved and to make him aware of the implications of what is known as the "hardware-software interface"—that is, the way in which programs interact with the computer to provide desired solutions in specific applications.

In general, there are two ways to approach the creation of a "program." One way is to assume that a particular computer with a given configuration of equipment is available: how should it be programmed for the desired application? The other approach would be to define the problem and then to determine what combination of hardware and software would best meet the user's needs. The latter implies a "systems" overview, and it assumes that the application is big enough to justify the installation of the ideal system. For the most part, "programmers" are handed assignments to perform on a given configuration of equipment. "Systems analysts" are asked to recommend the best overall solution, in terms both of hardware and software.

But for the moment, as a starter, let us consider only programming itself, more or less independently of any hardware implications. It is customary for a programmer to start out with a precise definition of the assignment and an identification of its component elements, which is usually accomplished by "flow charting" the steps which will have to be taken.

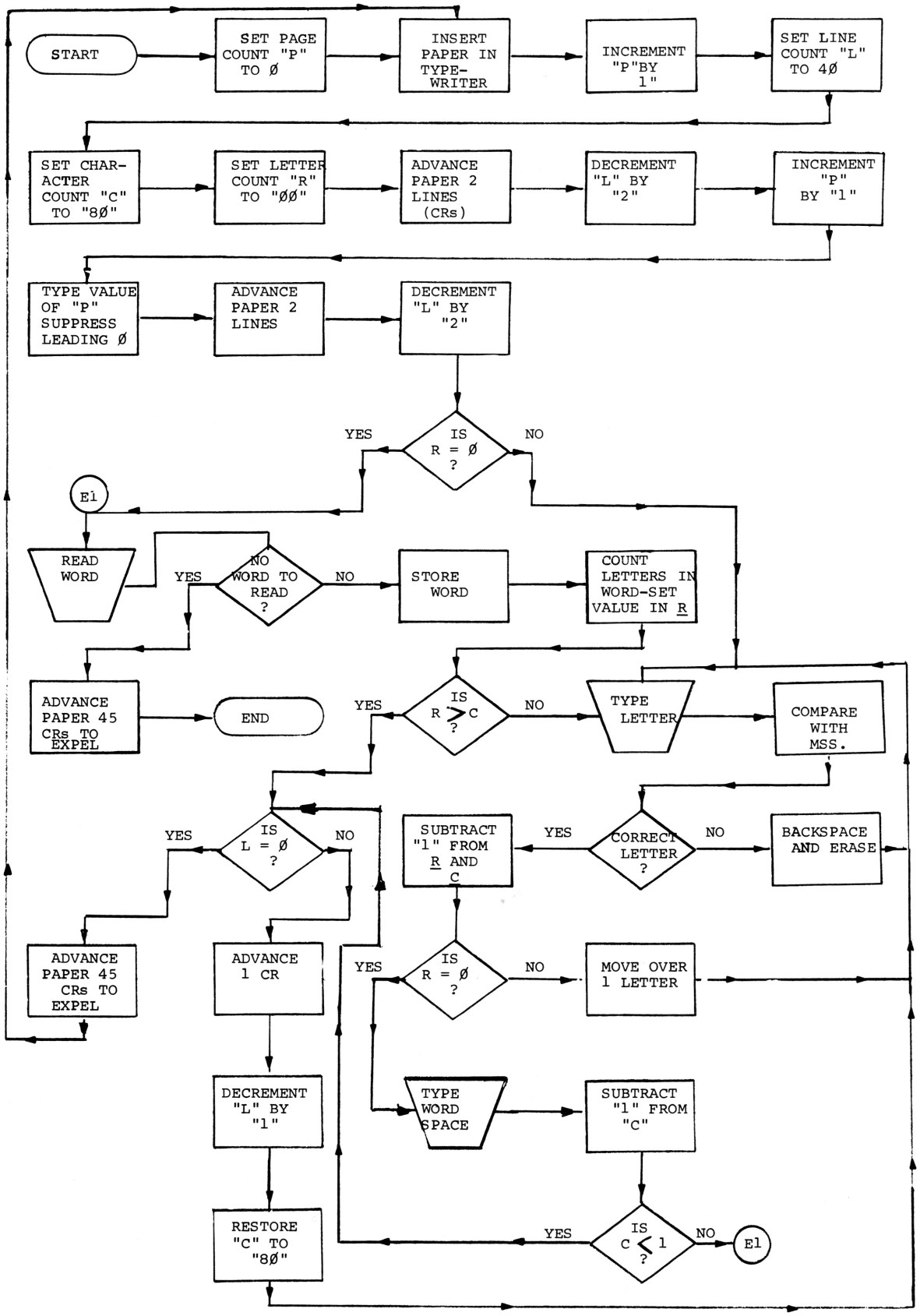
The reader will quickly discover that the flow chart is a most valuable tool, and if he can make himself "think like a computer" he will gain new insights into many problems and will improve his ability to analyze assignments and reduce them to manageable proportions. In many situations where the installation of a computer has been contemplated, potential users have discovered that the discipline required for computer programming—namely to break the task down into its relevant elements—produces results which are valuable in and of themselves and may even obviate the need for the computer system.

In order to illustrate how a flow chart may be prepared and what use it may be put to, we shall take as an example the task of typing a finished paper from an author's manuscript. We shall assume that the paper will be typed manually, but that the steps to be taken would be precisely those that a computer might be asked to follow. We will find, in fact, that the computer will be called upon to make the same decisions that an individual would make, although he generally makes them subconsciously, just as he walks without thinking: "Now I will advance my left foot; now I will shift my weight from my right foot to my left foot; now I will advance my right foot; when I

Printed in U.S.A.
GX20-8021-2 U/M 050
Reprinted 12/69

# IBM Flowcharting Worksheet

Programmer: SEYBOLD Program No.: 1 Date: 9/20/70 Page: 1 of 1
Chart ID: ____ Chart Name: TYPING MANUSCRIPT Program Name: SIMULATE TYPING MANUSCRIPT



advance my left foot I will swing my right hand backward to provide balance and momentum."

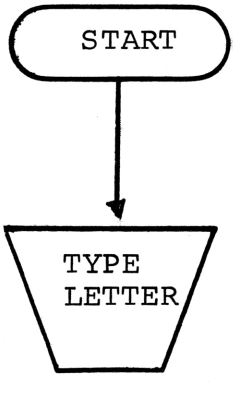
If the reader will study the flow chart carefully (and he may find errors in it, although we are not aware of any, at the moment) he will see that it consists of several different routines, which may be described as follows:

Start job and set up counters
A page numbering routine
A line counting routine which ejects the page when there is no longer any room on the page for additional lines
A word reading and typing routine, which continues to read and type words until all words have been read, and which types words character by character, until the next word will not fit in the line, in which case it is placed on the following line.
A letter verification routine

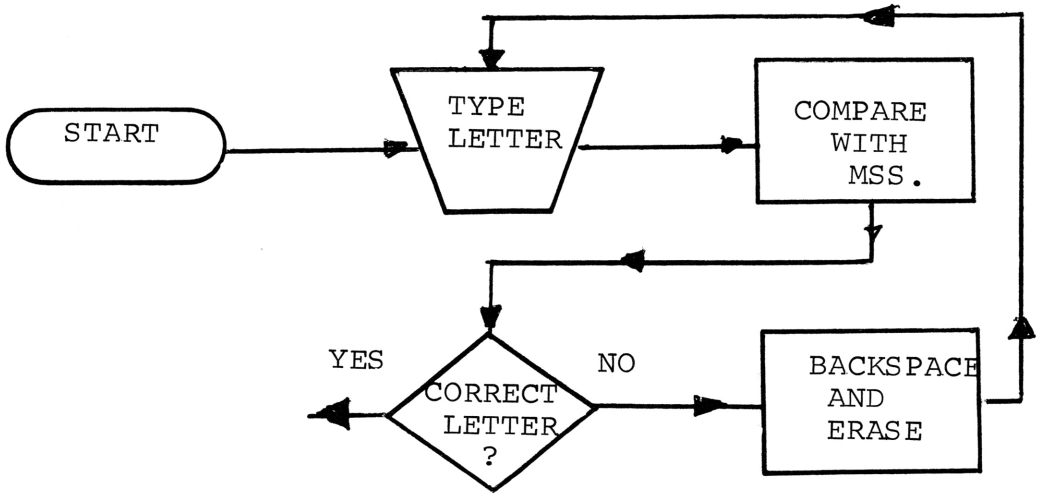
These routines are interlocked. In a sense, the program works as follows:

Start job; number page; go down to where the first line of copy should appear. Read a word. Determine if the word will fit in the line. Type the word, verifying each letter. Read a new word. Determine that the new word will fit on the line. Type the word, verifying each letter. Read another word. Determine (for example) that the word will not fit on the line. Determine that there is room for another line on the page. Advance typewriter carriage. Type the word, then read a new word. Determine now (for example) that there is not room on the line for a new word. Determine also that there is not room on the page for a new line. Eject page. Start new page. Number page, etc.

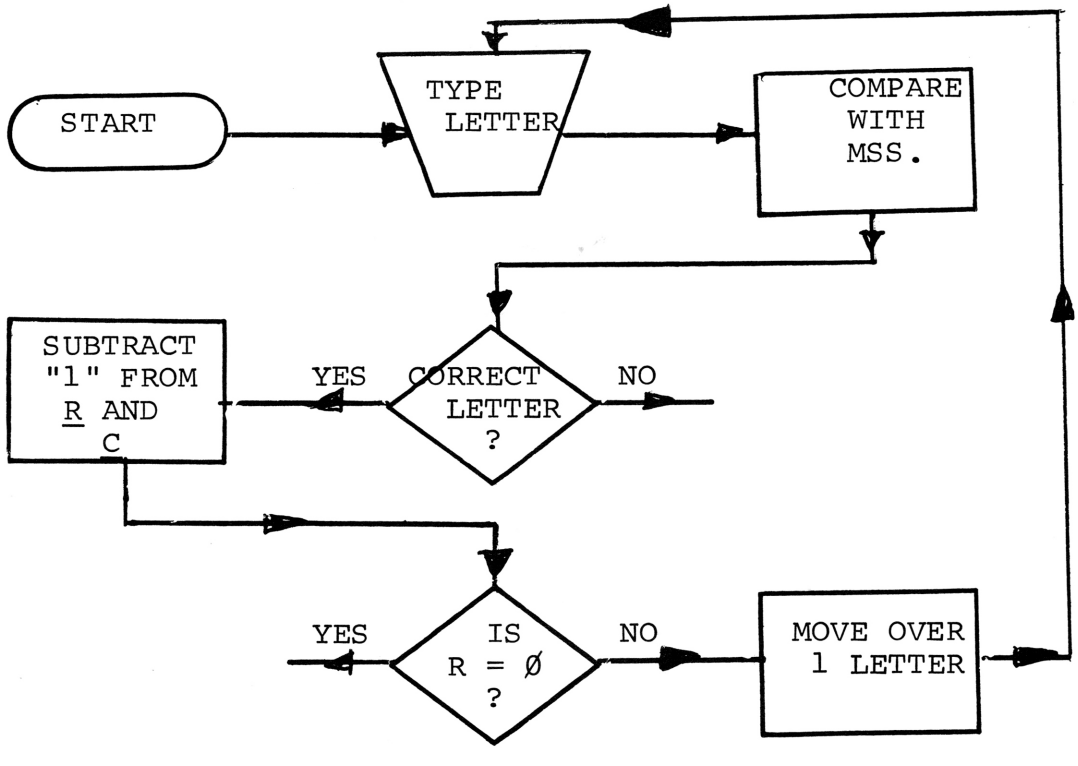
The reader may be confused with the evident complexity of the flow chart shown on the accompanying page and it may therefore be easier for him to start with something just a little simpler. Instead of starting at the beginning of our program, therefore, let us consider for the moment the typing of an individual letter. The shape of the first box (a keystone shape) will suggest that this is an "input" or "output" function. In this case "type letter" may be assumed to be an output function.



Now, suppose we imagine that a person types that particular letter (we mean character, of course) and then looks at it, and asks himself, "Is it the correct letter?" If he finds that it is not, let us assume that he backspaces, somehow rubs out the character and types it over again. Again he checks to see if it is the right letter.




You will notice that the "compare" function leads to a decision. In flow charting, decisions are indicated by a diamond whereas a rectangle indicates some kind of machine processing activity other than input/output or decision making. You will also notice that we have diagrammed a "loop." As long as the wrong character is keyboarded and detected, the loop persists. If the right character is struck, there is a "branch" away from the "loop." In a sense, then, this "loop" could be called a "subroutine."



Suppose we have a counter ("R") to indicate the length of the word we are typing, and another counter ("C") to indicate the length of the line. Whenever the correct letter has been typed the line length counter will be reduced ("decremented") by one, and so will the wordletter count for the individual word ("R"). When the wordletter count is at zero the word has been typed and a word space is needed (and that value has to be subtracted from the line). Until the wordletter count reaches zero additional letters in the word will be typed, and this makes another loop. We branch away from this loop when the wordletter counter is less than one. (It could be zero or minus one, since we have added a space after the word, even at the end of the line.) In this case we would go to the "new line" routine.

We need not belabor our illustration any further. The reader may wish to try a flow chart of his own on some specific task, such as backing the car out of the garage:

Do I have the keys to the car?
(*Get key* routine if answer is "no")
Is garage door open?
(*Open garage door* routine if answer is "no.")
Open car door
Get in
Insert key
Press starter
Put gear in reverse
Depress accelerator

Other sub-routines may be relevant. For example, should there not be a routine to inquire if gear is in neutral before pressing starter? What about releasing the hand brake? Some people would insist that the first steps after entering car would be to fasten seat belt, adjust rear-view mirror, etc. But did the reader notice one significant omission? Our flow chart does not include a very necessary instruction: close door of car!

Flow charting may be carried on at a very detailed level or on a very general level. Once a particular routine has been thought through, thereafter it may be sufficient merely to describe that routine within one rectangle, even though it contains many branches and offers several loops of its own.

One very important aspect of flow charting is to check continually to be sure of the appropriate conditions. For example, it is easier to include an instruction to see if the linecounter will permit the next word to be typed, even if we know that we have just gone to a new line and the word would most surely fit. This enables us to use the same loop or sub-routine for all words in the line, and not merely the first. And it also will bring to light the rare case where we define the line as being exceedingly short and encounter a word that is exceedingly long. What will we do if we find that the "word" contains more characters than the line length we have requested? This could happen either because we have mistakenly specified the line length as too short (perhaps we have misplaced a decimal) or some abnormality has been associated with the "word." (Perhaps we have dropped a word space, or we have incorporated a lengthy mathematical equation into the text.)

We can see, then, that our flow chart must consider every possibility, even the absurd or unlikely ones. We come upon the general principle of programming that we must anticipate every contingency. We must learn to see the conditions we are likely to encounter as generalities, and we must ask ourselves to visualize every possible

condition we might encounter, even for applications that we do not presently have in mind but for which someday we might wish to use the same routine.

### From the Flow Chart to the Coding

Once we have developed satisfactory flow charts we need to reduce our concept or plan of action to a specific computer program. This may be done merely by "coding" if our charts are specific enough. But if we have left open many questions and choices, then an experienced programmer—much more qualified than a "mere" *coder*—will of course be required. How does the program get written?

Ultimately our program must be reduced to machine language coding: commands that will instruct the computer in the specific performance of its tasks. If we wish, we can write our program in machine language, although this is not done very frequently. When you write in machine language you have to keep track of every single location of every element you are working with. If you "read in" a data element and decide to store it temporarily in location 1000 then you must remember that that is where it is, and you cannot use this location for any other purpose. You must also plan your program so that all of your instructions are sequential. For example, if all instructions take ten bytes, your first instruction would begin, presumably, at location "all zeros" and your second instruction would be brought in next as the computer automatically called up location 00010. If you wanted to add a sub-routine between the first and second instruction you would have to re-number and re-assign all of the subsequent instructions.

Consequently, you are not likely to write programs in machine language. The closest you would come to machine language would be "assembly" language. Here you will use many of the same instructions and codes, but a program will later be run against your program, and it will assign specific core locations to your instructions and your data areas. When you modify your program these locations and areas will be reassigned as needed. The computer manufacturer will supply you with an "assembler" which will take your "subject" program and create an "object" program in machine language. Each time you reassemble it you will derive a new object program.

Your assembly language program will not be very "portable." It will be written in a language which can only be understood by the "assembler" designed for the particular make of computer. If you want a more "portable" program you will want to write it in a "compiler" language, such as FORTRAN, COBOL, ALGOL, or perhaps BASIC or PL-1.

The computer manufacturer will presumably have provided you with a "compiler" which will translate the programs you write in FORTRAN, for example, into machine language for your computer. Your subject program will be "compiled" into an object program in machine language, and re-compiled as necessary, whenever you revise it.

When you use these higher level languages for your programs, then you pick up many existing sub-routines for certain basic functions. Programming is usually much simpler, quicker and more trouble-free. You do not need to "de-bug" routines which have already been perfected by others. On the other hand, the routines others have written for you may not do precisely what you want to do, and you may have to achieve the desired results by more roundabout procedures. Specifically, for typesetting applications, compiler programs are usually not efficient, if they will work at all. Such programs are usually intended for the solution of problems in business economics, or mathematics, or science, rather than to do character-by-character processing,

such as is usually required for typesetting applications. Thus the kind of programming generally taught in colleges and universities and which is designed to help a businessman or a scientist gain access to a computer to do occasional tasks, just as he might sometimes need the services of an adding machine, may not be very helpful where "production" programs are needed which must be quick and efficient.

### Loading the Program

When a program has been written and debugged, so that it is available for use in a production environment, it will generally be stored in its machine language format, and it will not be necessary to re-compile it or reassemble it each time it is to be used. The stored program may be kept on magnetic tape or on punched cards, or it may remain on disk. When it is to be run, that program or a relevant portion of it is called into core, sometimes by operator intervention and sometimes by a terminal command from the user. The program sets up its own registers and establishes whatever conditions are needed, or checks to see if the relevant pieces of peripheral equipment are ready and available. It then accesses the first instruction, executes it and moves automatically either to the next sequential instruction, or, if there is a branch or conditional transfer, the program contains the address of the location of the next sequence of instructions. Hence the program is in "charge" and the computer operator's function (if everything turns out as it is supposed to) is merely to service the requirements of the computer, such as to mount or take off tapes. If the program "hangs," warning lights usually appear at the computer console or the operator is otherwise notified. This is not invariably the case. Sometimes a program can get into a loop, from which it can never extricate itself, and the operator must be aware of probable running times and other expected conditions so that he can diagnose problems if they seem to occur. If something goes wrong he can abort the run or try to recover, and usually will be instructed to take a "memory dump" and readings of registers so that the programmer can try to find the source of the difficulty at his convenience.

Operator diagnosis or intervention is not likely to be helpful where "multi-programming" or "multi-processing" is taking place. This occurs when the same computer is able to run several different jobs more or less simultaneously. Multi-programming describes a condition in which each user's program works within a "partition" of core, and time-sharing describes a condition under which various users share the same sections of core in successive very brief slices of time. Under either of these conditions, or combinations of both, all programs run under an "operating system" and the operator's attention is directed primarily to the requirements of the system rather than the peculiarities or shortcomings of individual programs. It must generally be assumed, therefore, that successful computer programs must be trouble-free, and they must anticipate all conditions to which they might be exposed. They must also be designed to provide recovery points so that a job which aborts need not be started again at the very beginning. Diagnostic features can and should be built into the programs as well as ways of keeping track of the status of the job.

### Documentation

Flow charts provide the basic clue to an understanding of the program, but a more detailed explanation is essential, and such documentation can be provided by the programmer as he does his coding. Explanatory comments may accompany his array of instructions, and these in themselves, if produced conscientiously, may be sufficient to enable another programmer to understand what the first has tried to do. As an

illustration of IBM 360 assembler coding here is an example of a printout which shows the "object" or machine coding (the first five columns) generated from the source coding by the IBM 360 assembly program, from assembler coding. "STMT" means *statement number.* The actual assembler coding consists of the data to the right of the statement number. "S4LOOP" is a tag assigned to a sub-routine.

```
 LOC    OBJECT CODE       ADDR1  ADDR2  STMT          SOURCE STATEMENT
000790  4170  B084               020A4  602             LA    R7,TABLST     POINT TO FIRST ENTRY
000794  5860  C070               01090  603             L     R6,NUMTAB     GET TABLE COUNT
000798  5850  C074               01094  604             L     R5,TABADD     AND ADDRESS OF AREA
00079C  9500  7002        00002         605  S4LOOP     CLI   2(R7),0       R1 OR R2 FAMILY?
0007A0  4770  D7B6               007D6  606             BNE   S4SKP         SKIP IF NOT
0007A4  D200  505D 5000   0005D  00000  607             MVC   X'5D'(1,R5,0(R5) MAKE EN FIGURE WIDTH
0007AA  4155  0080               00080  608             LA    R5,128(R5)    TWO TABLES IN R1
0007AE  D200  505D 5000   0005D  00000  609             MVC   X'5D'(1,R5),0(R5)  DO IT FOR R2 ALSO
```

LA means "load address"
L means "load"
CLI means "compare logical immediate"
BNE means "branch not equal"
MVE means "move character"

The tags which follow these instructions designate registers or routines. The final column contains explanatory comments by way of documentation.

This particular piece of programming was taken from a routine used to calculate width tables.

### Review Questions

1. What is a "systems analyst"? A "programmer"? A "coder"?
2. What is a flow chart? How detailed must it be?
3. Select some pattern of activity and see if you can draw a flow chart of all of the steps involved. If you are a linotype machine operator, for example, you might try to describe your activities in such a chart.
4. What is "machine language"? "Assembler coding"?
5. Why is a "compiler language" more portable?

CHAPTER X

# Software Elements For a Typesetting System

Certain of the elements which will be discussed in this chapter are essential to any computerized typesetting system. Others are features which may appear only in the most advanced systems and would be irrelevant for crude applications, such as hot metal typesetting, where the output device (i.e., the typesetting machine) imposes such restraints upon the system that refinements like computerized page makeup would not be useful.[1]

We will begin our run-down of possible computerized typesetting systems by considering the most simple and most widespread application: this is the so-called "one-pass" system. A single computer pass processes input through to final output without interruption for human review or intervention. Although several programming routines or sub-routines are involved they are all part of the same general program, the purpose of which is to accept an unjustified stream of text and to produce a set of output codes which will cause a typesetting machine to compose justified and hyphenated lines in galleys of job length.

A multi-pass system, on the other hand, contains several "programs," each consisting of many routines or sub-routines. These programs are often called "phases." They are run consecutively, although some of them may be run several times, and some of them may be run in different sequences. Multi-pass systems are so devised for two reasons:

1. because they are too big to fit into core as one program, and/or
2. because human intervention and review is desired at different stages in the production process.

It is this latter point which is the most significant, since, if such review were not required, programs too large for core could be run under computer control as if there were one continuous production process, by overlaying one routine or phase on top of another, provided some means of interim storage of the partially processed product can be devised.

A very sophisticated version of multi-pass programming permits the processing of jobs in minute pieces subject not only to the possibilities of human review upon the

[1] There is, apparently, a hot metal page makeup program, however. This was reported by (Mrs.). S.R. Stewart, of Elliott Automation, at the 1969 conference at the University of Newcastle-upon-Tyne.

termination of each phase of the production run, when the "batch" of material has been processed through that run, but also to facilitate intervention at virtually any stage within the run, in order to review and modify the input or the results of the process to that point of time. Such programs are called "interactive" because they respond in "real time" to the user's demands. (Real time means instantaneously, or with virtually no perceptible delay, so that the thought and action processes of the individual are not inconvenienced.)

### The One-Pass System

We shall begin, then, with an explanation of a one-pass batch processing system, and for the moment we shall assume that the computer is wholly dedicated to this operation and that it functions under operator control. The operator decides when the program is to be run, runs only that program at that time, loads the program, mounts the input, and takes off the output. For the most part such programs use a relatively small and inexpensive computer, and the program may be "input bound," "compute bound," or "output bound."

If the program calls for the reading in of paper tape, the speed of the program may be limited by the speed of the paper tape reader. If it calls for the output of punched paper tape, the speed of the program may be limited by the speed of the output paper tape punch.

Let us suppose, for example, that the input paper tape reader is capable of reading at the rate of 200 characters per second, and that the paper tape punch is only capable of producing the final punched tape output at a speed of 100 characters per second. Let us suppose, further, that the computer calculations are faster than the punch can handle. We would then expect the reader to read intermittently, the computer to "compute" intermittently, and to service the output punch most of the time, so that the latter runs continually.

Let us suppose, on the other hand, that the paper tape reader can read at only 100 characters per second, and that the punch can produce 300 characters per second. In this case, if the computer processing speed is not a limiting factor, we would expect the paper tape reader to be working away constantly, while the punch functions intermittently. An inbalance between reader and punch, in terms of relative speed, would appear to be undesirable from a "systems" point of view, other things being equal.

But let us suppose that the computer program which performs the necessary computations and logic steps is either poorly written or very complicated. In that case the program may be "compute bound" and perhaps both the paper tape reader and the paper tape punch will run intermittently, awaiting instructions from the computer while it is busy digesting what it has already ingested.

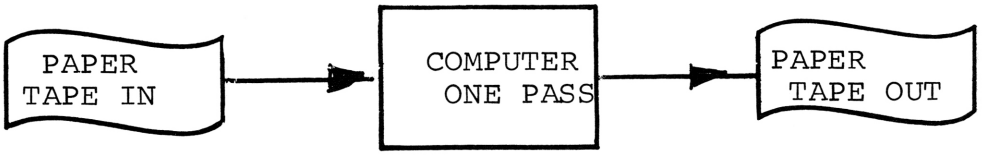
In a one-pass, batch processing system which does not operate under "multi-programming" [2] and where the computer is not called upon to perform other functions (as, for example, payroll), then the economic justification for the installation would have to depend, of course, upon the volume and efficiency of the work it performs, as contrasted with pre-installation composition techniques. And the efficiency of the system would depend upon the "throughput speed" of the one-pass application.

[2] "Multi-programming" permits the same computer to perform different tasks simultaneously, in different partitions of core.

Assuming paper tape input and output, the essential routines of the one-pass system could be described as:

- read input
- certain diagnostics and housekeeping routines
- translate input into an internally compatible language
- perform justification function, accessing whatever tables of character width values are relevant
- perform hyphenation function as required
- interpret character strings with respect to the function and command requirements of the output device, as well as its text stream code structure
- output paper tape in this code structure

Processing will occur by reading in a string of codes which will fill up a work area (perhaps 200 or 300 input characters, for example). When these characters are output (or most of them) additional characters will be read in and processed. A certain amount of overlapping would be possible, so that new characters can be input while old ones are being outputted. The following schematic will be adequate to describe the essential features of the one-pass system.




We shall not endeavor, at this juncture, to particularize what takes place within the computer box since this will be the subject of much more detailed analysis in subsequent chapters. The "justification" function may merely consist of bringing each line *within justification range,* rather than to put out the specific amount of interword white space necessary to justify. The typesetting unit itself may perform the necessary justification, either by spaceband expansion or by the kind of simulation of spacebands which first or second generation photo units are able to perform.

A single pass system generally assumes that no correction will take place. If there are errors in input they will appear in the output. Once discovered, it may be possible to go back and improve the input (perhaps by splicing in a new section of paper tape) but the probabilities are that most of the inadvertent errors, or other desired revisions, will be made by conventional methods, such as removing one or more hot metal slugs and substituting new ones which have probably been produced without the assistance of the computer.

The absence of a correction capability is not a significant handicap when it comes to hot metal typesetting, but one-pass systems are less attractive for photocomposition, since corrections are harder to make from film or paper than from hot lead. One would therefore expect that the bulk of computer programs for hot metal typesetting would be of the one-pass variety. It is surprising to note, however, that there are also a large number of such programs being used for photocomposition. In many cases it is

the same computer program, initially designed for hot metal setting, that has been modified to output to a photocomposing device. In other cases, entirely new programs have been written. The lack of a correction capability, in the opinion of some, is offset by the convenience of keyboarding input to a photocomposition device via computer rather than by direct access. It is sometimes cumbersome to provide the necessary keystrokes to put the photo unit through all of its many and varied paces. Computer programs may do a better job of counting character widths for the setting of mixed composition than can be done by the use of justifying keyboards which would have to access many different combinations of width tables.

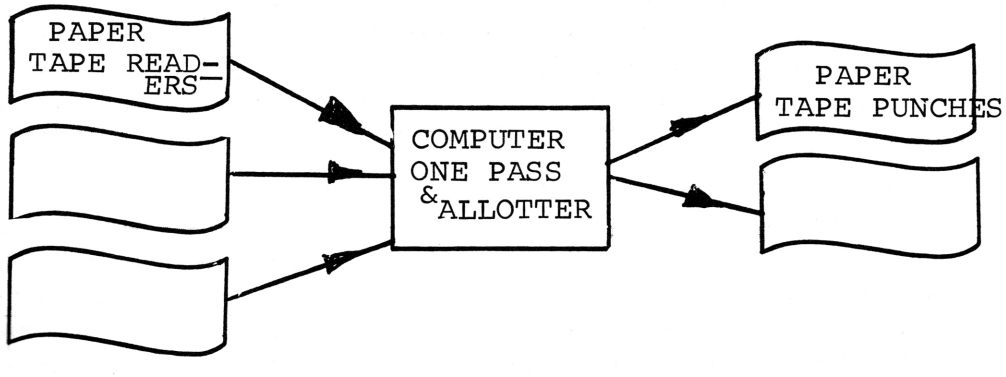
### The Allotter Function

One interesting variant on the one-pass system is its use in the performance of an "allotter" function. For hot metal linecasting composition especially, and notably in newspaper composing rooms, it is possible for the computer to perform the task known as "copy cutting."

"Copy cutting" is a term sometimes used to describe the duties of a markup man or a supervisor who glances over the incoming manuscript and notes that certain portions are to be set to one measure and certain portions to another. Certain "takes" are to be set in one type face and certain ones in another. There are differences in type size as well, and there may even be differences in the character repertoire available on certain typesetting machines. Hence the copy cutter assigns different portions of the same manuscript to different operators, even by cutting pages into appropriate elements. After the material has been typeset, usually before proofreading, the manuscript is reassembled, and all of the material is rearranged in proper consecutive order.

But with a computer system it would be possible to give the manuscript to one operator. He could key the entire manuscript, including codes which would allocate the output to different output devices. The paper tape output punches can be located near the various linecasting machines: the wide measure ones, the mixers, the machines that are adjusted to produce a certain measure or are equipped with magazines that customarily cast a certain size. The computer program still reads only one batch of paper tape at a time, but it will allocate the output to different punches, only one of which will be busy at any given moment. In fact, several paper tape readers can be employed, and when the keyboard operator has finished a take he can signal the computer that he has material to be read. When the computer has finished processing its current batch it will take the next assignment from the queue, or with whatever priority allocations have been programmed.

This diagram illustrates the use of the one-pass system in an allotter environment.



Of course the various typographic elements will have to be assembled manually, by conventional means, even though they are input as one entity, since they have been separated out and set on different typesetting machines.

It can be seen that this allotting function is able to contribute a great deal of utility to the composition process since it is able to perform the function of copy allocation efficiently and promptly.

### Handling Corrections in a One-Pass System

If a way can be found to proofread the input prior to typesetting, it may be possible to improve that input and get a better level of output. There are three ways in which this can be done. One is to proofread the hard copy produced by an input keyboard. A second is to obtain an off-line tape print, which provides the equivalent of hard copy. A third method is to secure a computer printout of the input tape.

Some input keyboards provide hard copy. Of course if the keyboard operator discovers his mistakes immediately upon keying them he can rub out the faulty keystrokes and insert the proper ones. But if they go undetected it is still possible to read the hard copy and see what it is that he has punched onto the tape. The trouble is that very few hard copy representations are able to describe all of the keystrokes which appear on the tape. There may not be hard copy equivalents for certain kind of spaces (such as ems, nuts and thins) or for quad commands (left, right and center) or for font shift commands (up or down the rail). Moreover, where deletions have been made by rubouts it is not always possible to detect which are the original keystrokes and which are the corrected ones. Nevertheless, there is some practice in the industry of reading from this hard copy and detecting errors before the paper tape has been given to the computer. How then do corrections find their way onto the revised tape?

If one can find his errors on the paper tape itself he can cut and splice. But for any quantity of corrections this would be awkward. A keyboard device which contains a paper tape reader and a reperforator unit is sometimes used. The operator plays back the original tape and where it is error free a new tape containing the same keystrokes is produced. Where the first error occurs the tape reader is shut off and the operator keys the desired corrections. Then the tape reader takes over and continues to the point of the next correction.

A similar paper tape merging capability can be created by obtaining a printout from a "blind" keyboard by means of a Teletype printer device. This printer can have the capability of reperforating, and it too can be interrupted to accept input from another keyboard or from another paper tape reader.

More interesting is the possibility of adding an on-line printer to the computer configuration. Then the paper tape can be printed out when it is read in. Every single character on the input tape can be represented by some graphic symbol. Lines can be numbered so that the proofreader can indicate the area where the correction is needed, and then corrections can be inserted in a manner which we shall discuss shortly.

### The Computer Printer

Many computer printers have a limited character repertoire. Some of the early ones use a "print chain" of only 44 symbols. Others offer 64 characters, but all of the characters of the alphabet are uppercase only. More recently a larger character repertoire has been made available (at more money, of course) so that upper and lower case characters can be shown. A computer printer does not have the capability of printing

certain character strings in red, but it can represent all of the codes so that what is on the input tape can be divined with little difficulty.

```
+#H&ERE IS AN EXAMPLE OF AN UPPER CASE PRINTOUT./
```

In this case we have used a "plus" sign to indicate an em space. The number sign represents an upshift code and the ampersand represents a return to lower case. We have used the slash to mean that a quadleft code will also be present.

```
+Here is an example of an upper and lower case printout./
```

If we need a function key to indicate a shift, as to italic, it might be represented by an asterisk, and a second symbol could be selected to represent a shift back to roman.

```
#N&OW WE GO TO *ITALIC ¢AND BACK TO ROMAN./
```

Or it may be that two keystrokes are used to signal italic and roman. If the asterisk is followed by an "R" we can assume roman, and if it is followed by an "I" we can assume italic.

```
#N&OW WE GO TO *IITALIC *RAND BACK TO ROMAN./
```

If we are using an upper and lower case printer:

```
Now we go to *Iitalic *Rand back to roman./
```

### How Corrections can be Processed

A computer printer can operate as a peripheral to the computer or it may be run off-line. However, in the context of this discussion it would probably be on line. Assuming it is so used, and that we read in an imperfect paper tape, we now have two options available to us:

1. we can get a tape print by computer printer of the input tape, broken more or less arbitrarily into lines;
2. we can perform the functions of hyphenation and justification, and get a print of the results—either outputting the product simultaneously to paper tape, or postponing the paper tape output until the printout has been read and corrected.

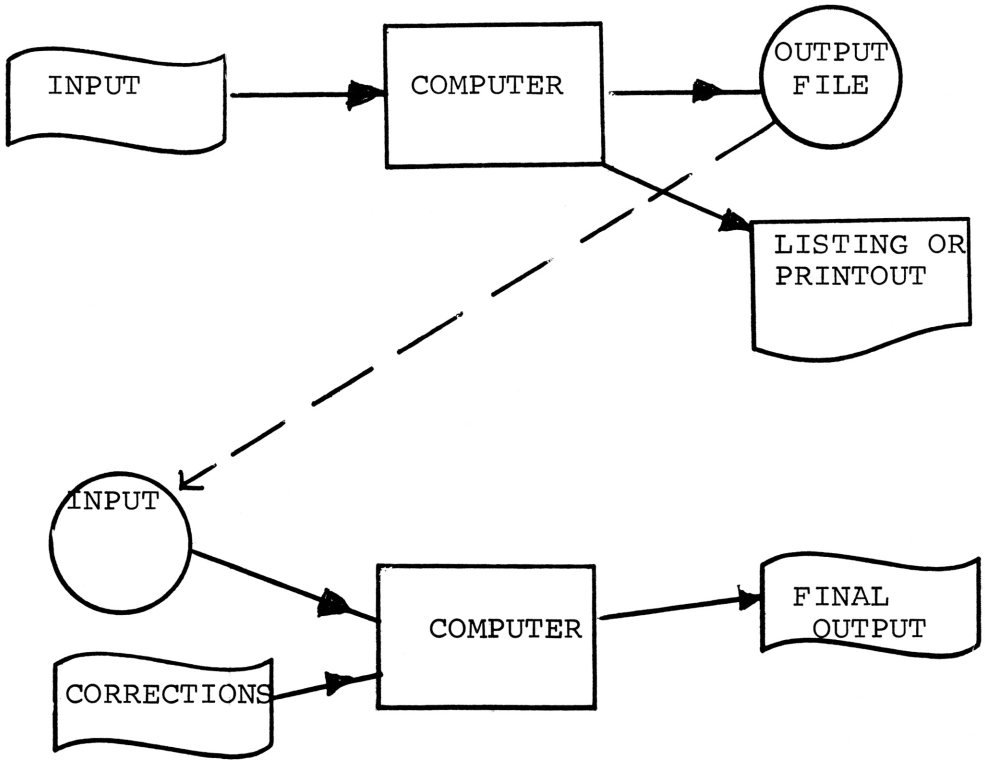
In either case, however, we have to correct *something,* and where these techniques are used a correction technique somewhat more sophisticated than a paper tape merge is generally employed. One of two alternatives can be followed:

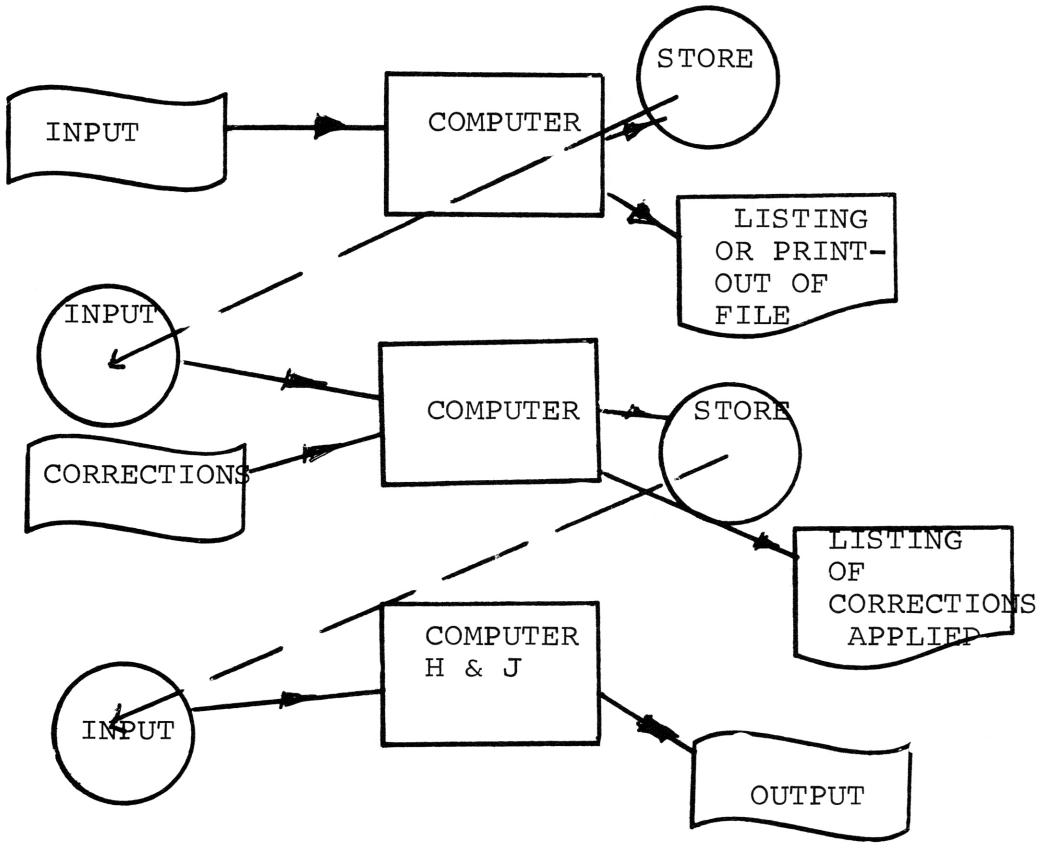
1. a paper tape containing only corrections can be created. This tape can be read in and somehow stored. Then the initial tape can be re-entered into the system and the corrections, already in store, will take the place of the invalid keystrokes.
2. the entire input can be stored somewhere on line to the computer (magnetic tape, disk or drum, since presumably there would not be room in core) and then the corrections can be entered and a new file created. A printout could then be obtained of the new file. This process can be repeated as often as desired.

**Multi-Pass Systems—Corrections**

Whether the first or the second alternative is chosen, we have now moved to a multi-pass system. The second pass is the correction pass. It may be the final pass, since it may also be the output pass, as is indicated by the first of the two methods just mentioned. It may not be the final pass. Corrections may be incorporated and then a final or output pass may be produced.

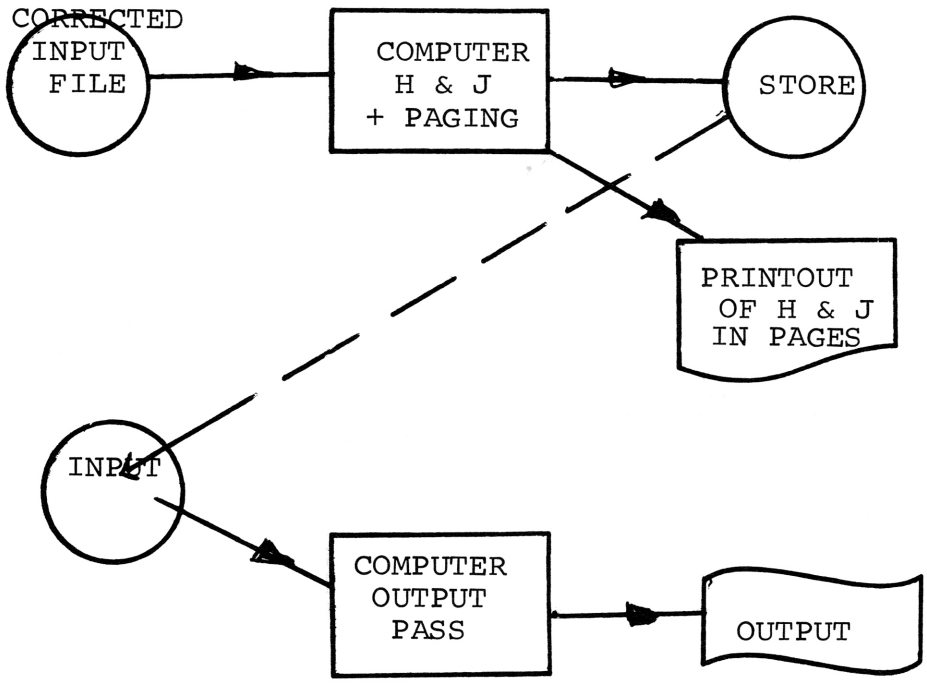
Here are diagrams showing various alternatives:





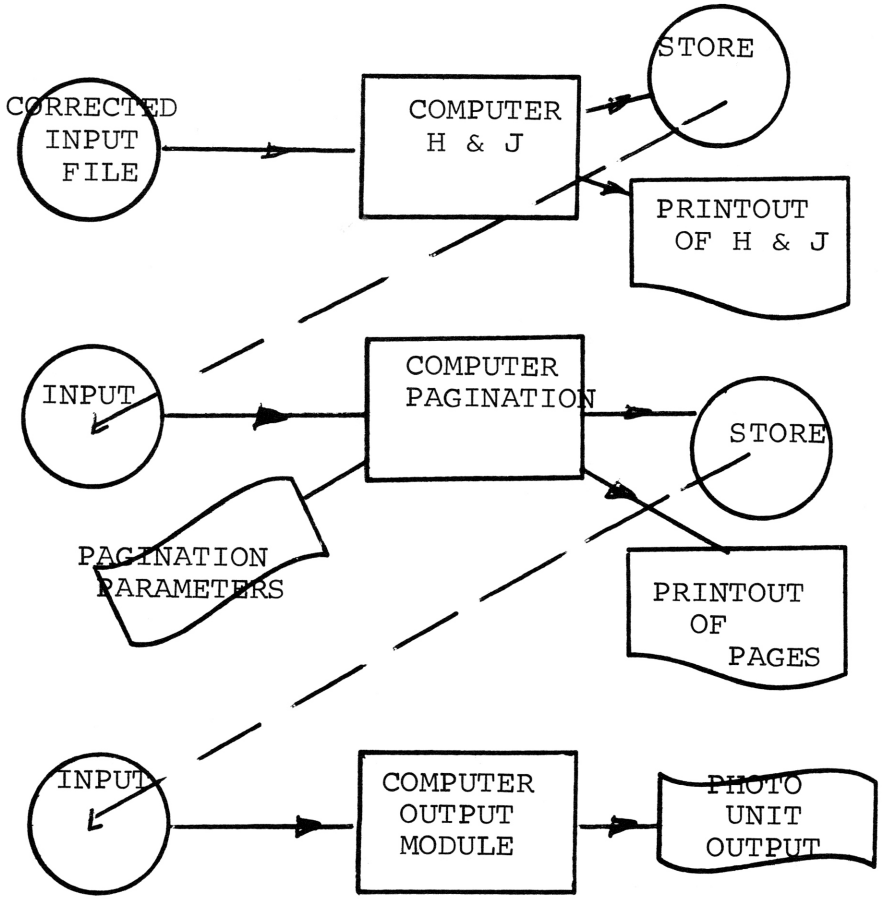

**Pagination**

Page makeup, or pagination, is also a function that can be performed by computer. There seem to be two ways to accomplish this. One is to combine pagination with hyphenation and justification. In such a system lines of type are composed in pages



and running heads and folios are program-generated. Thus no additional computer pass is required, but additional demands are placed upon the hyphenation and justification routines since they become part of a larger composition program.

Alternatively, the output which has been corrected and has been run through hyphenation and justification can then be processed by a page makeup program.




**Separate Output Modules**

There has been an increasing tendency to keep the interim file in a generalized format. When this is done a separate output pass is written for whatever typesetting machine is desired. While such a pass could be merely a tail-end module of the pagination routine it may also be a separate computer run. The output pass should be able to accept as input either galley material or a file that has been paginated.

### Separate Input Modules

There has been an increasing tendency to disassociate input from the typesetting package itself. This is because input may be derived from a number of different sources and each source may have its own peculiarities. Different kinds of paper tape devices may be used, with different code structures. Some of these devices may produce tape that is not compatible with the paper tape reader. Some of them may therefore have to be converted to magnetic tape off line. Some input keyboards may not produce paper tape at all, but may write directly to magnetic tape or to a computer disk file. Some input may have been produced as magnetic tape by another computer. Consequently, there may be a choice of computer passes or off-line manipulations in order to develop input which is processable into the composition system. We have now suggested a *modular concept.* There can be several input modules, one or more basic composition phases, including correction capability, and several output modules.

### Multi-Programming and Time Sharing

The computer programs we have discussed may run on a wholly dedicated system or they may run at the same time that other programs are being run. If the computer configuration permits the use of an "operating system," then these batch programs may perhaps be run in a multi-programming environment. Thus at the same time that input is being handled by one partition of the computer core, hyphenation and justification may be in process in another partition. Or payroll may be being calculated while a typesetting output pass is running.

It is even possible for typesetting programs (if so intended) to be run in a time-sharing environment. In such a case it is the task of the operating system to "swap" core as between users so that portions of each task (typesetting and otherwise) can be produced more or less simultaneously. One significant difference between multi-programming and time sharing is that in the former each job is produced in its own partition until it is completed, whereas in time sharing a program is loaded only for an instant, a small assignment is carried forward and then another program, another job and another user are serviced with the same core and peripherals. But this happens so fast that on the whole no user is aware that others are sharing the equipment, which he may imagine to be wholly dedicated to his application. Of course, multi-programming and time sharing may proceed at the same time. In other words, a time-sharing application may be multi-programmed, or a multi-programming application may be time shared.

These sophisticated applications usually occur only on "big" or "medium scale" computer systems. The cost of developing the appropriate operating system arrangements is very great, although they are frequently supplied by the hardware manufacturer. User programs may not be able to be run under time sharing, but most user programs *can* run in a dedicated multi-programmed partition. Later in this study, where we are concerned with markets and with economics, we shall revert to these highly complex systems applications. But for the moment it will be necessary to move next to an examination, in considerably more detail, of the essential features of a composition system, and in particular we need to look at hyphenation and justification, corrections or "editing" and pagination.

**Review Questions**

1. What is a one-pass system? What are its advantages? Its disadvantages?
2. What kind of typesetting system would you recommend for a hot metal application?
3. Under what circumstances would an allotting system make sense?
4. Why would allotting be of less use in a photocomposition environment?
5. What function is served by adding a computer printer to the computer configuration?
6. Why would the addition of magnetic tape or disk storage add to the computer's typesetting capability?
7. What are modular programs?
8. Why should you consider a system which permits access to more than one output device when the justification program has to cope, in any event, with the width values for a particular typesetting machine?

CHAPTER XI

# Justification Examined

It will make no difference, in our discussion of this subject, whether we assume input from paper tape or from magnetic tape. The computer will be instructed to accept what amounts to a continuous stream of data, although if the input is magnetic tape it would be "blocked" so that there is a tape gap after a certain character count. The stream of codes which will be read and processed, in either event, are basically of two kinds: instructions to the computer, and text material. The first category consists primarily of "tags" which identify typographic "situations" or conditions, such as paragraph endings or the beginning and ending of display heads. It is the function of the computer program to be able to distinguish between such flags, tags or signals (which are intended to activate certain processing routines) and the balance of the input (which is the text matter that the instructions relate to).

Usually the first block of information to be processed has to do with setting up the job. Perhaps there will be a job title, or an operator number, or some other type of identification. This is useful for record-keeping purposes, but it may also be a key element in determining whether a later run is being processed against the proper file. (We shall use the word "file" from time to time to describe the store of text data and related codes either in process or available to be processed.)

After dealing with matters of identification, the justification program needs to know the line length for the job, or at least for the first element of the job. Lines can only be justified against a particular length or measure. The program also needs to know the kind and size of type that is supposed to fit in the line.

Immediately after the identifying material, the program will look for line length information, and let us suppose that it is expressed in picas and points. We shall imagine that a flag, which we shall call "gamma" for want of a better name (and this is in fact the name assigned by the *Time* magazine composition program) is used to signal to the computer that it is about to receive an instruction. Suppose we conceive that when the gamma, which we shall represent hereafter as an asterisk, is followed by an upper case letter M, the computer knows that the four digits which follow will express the line length or measure in picas and points. Thus *M2210 would call for a line length of 22 picas and ten points, where *M1300 would mean 13 picas.

Our first problem is to reduce this line length to units of measurement which will correspond with the set width values for the type characters we are going to try to fit into the line. We can elect to compute type widths in terms of an absolute unit system which is expressed in millimeters or thousandths of an inch, or even, perhaps, in tenths of a point. (Thus a six point em would consist, say, of 60 units, and a twelve point em would consist of 120 units.) We might, it we preferred, adopt a relative unit system, so

that an em of whatever size would be made up of 50 or 60 units, or some other number, like the 18 units per em count of the Linofilm system. In either case (relative or absolute values) there are special problems which will be encountered in the conversion of character widths to units, especially where different type sizes are intermixed in the same line. We shall revert to this subject at a later point in the discussion. For the moment, however, we shall assume that we have established a line length (13 picas) and a point size (8 points), and a unit count based on 80 units to an eight point em. The input tape may specify the desired point size, or the type face may merely be tagged with a label which is defined at the point of processing. Let us suppose that we use tag *R to indicate the roman text type face. Therefore the first information on the input tape, after the identifying blocks, would be *M1300*R. Simultaneously our program would be informed by a definition of "parameters" that *R would be defined, at least for the moment, as eight point Times Roman.

We shall imagine that the computer has been given this parameter definition, and has now read first the identifying blocks and then this much of the text stream into a work area of core which we shall designate as area "A."

```
*M1300*R+This is the first line of input

into our typesetting program.=
```

(We shall use the plus sign to indicate an em space, and the equal sign to indicate an end of paragraph or quad left code.)

The program has now been told that a 13 pica measure is wanted and that eight point Times Roman is the font that is to be used. The initial calculation which the program must go through, therefore, is to determine the number of eight point ems which fits the line. Since 13 picas equal 156 points (13 picas x 12 points), 156 ÷ 8 establishes that there are 19½ ems in the line. If an eight point em consists of 80 units, we know that the line will contain 19½ x 80 or 1560 units.

It will now be necessary to bring into computer memory one or more tables of width values. For the moment let us assume that we have somehow loaded into memory one table which establishes width values for the eight point Times Roman font in which we are going to set the text matter. Using an octal coding structure in order to keep our example simple, we may imagine a table which reveals that the code which represents an em (assume it is 010 000, or an octal 20) has a width value of 80 units.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | 80 | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

Consequently the program writes the code for an em space into another work area (B), and decrements the line counter by 80. It goes on to the next character, which is a capital "T." Since we have supposed an octal system, the program learns of the fact that the "T" is an upper case letter by the existence of a precedence code, which is an upshift. The program therefore goes to the upshift portion of the table, and learns that this capital "T" has a value of 60 units. It writes the "T" to area (B) and decrements the counter by 60, so that it now stands at 1420 units.

The table look-up procedure just described is more complicated than appears on the surface. Actually, the em value of 80 or the "T" value of 60, which is stored in the table or tables, is best represented by a single character. In other words, a letter "K" might be stored in place of an "80," and elsewhere there will be a subroutine which informs the program that a "K" is equal to 80 units. We have translated the em into a "K," in effect, in order to show that its value is 80. This "K" is not to be confused with the text letter *K*, which also might be translated into a "K" if its width value were in fact 80, but more likely would have some other width.

Other characters may have the same value as the em. An em dash, for example, would most certainly have this value. If the em dash is an octal 32, our table would look like this:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | K | | | | | |
| 1 | | | | | | | | |
| 2 | | | | K | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

In an octal system, if we represent widths by characters, we will be able to describe something less than 64 different width values, because the six bit configuration can only yield that many combinations. The chances are, however, that we would not need the first twenty values since they would be too low. Nevertheless, an *octal* system which uses tables and a "translate by table" instruction may have problems storing a sufficient range of values.

So far we are working on the first word within the line. If we define a "word" as consisting of all of the consecutive codes in a string up to the point where a justifying space or a quad is encountered, the initial "situation" codes and the em space are also part of this first word. If we are setting "This" we have now to look up the "h," "i," and "s." Suppose "h" has a value of 40, "i" of 25, and "s" of 40. We have now decremented the counter by an additional 105 and it now stands at 1315 units. Then we encounter a justifying space. Our program will have been assigned a minimum value for that space, and also a maximum value. Alternatively, depending upon how the program is conceived, it could be given a "normal" value. Let us suppose we have a program which must be told, for each job, what the spaceband ranges may be, and we have therefore instructed the program that the minimum interword space shall be 30 units, whereas the maximum value shall be 60. The program thus would subtract 30, the minimum value, from the 1315 remainder, leaving 1285. Also it would tally in a separate counter the fact that it has encountered one justifying space. It will then

proceed to the next word, *is,* and by the time it has processed the word and the following space our character counter would be down to 1190 and we would have tallied two justifying spaces. (Hereafter we will call these spaces "spacebands." Although they are not in fact mechanical spaces; conceptually they serve the same purpose.)

We proceed to deduct 30 for "t," 40 for "h" and 40 for "e" along with the third spaceband, reducing the counter to 1050. The word "first" might consist of 30 for "f" and 25 for "i" for a total of 55, but before we finish up the word we might realize that "fi" is a ligature, and if our routine is confident that these letters will invariably appear on the same line it might ascertain that the value of the ligature "fi" is 45. If "r" has a value of 30, "s" we know is 40 and "t" is 30, and so we have deducted another 145 units, with still another 30 for the fourth spaceband. We are down to 875. Let's suppose that "line" brings us down to 615 when we include the space following the word. Our spaceband counter now stands at five. When we include "of," and its space, we have used six spacebands and 515 units remain. The word "input" takes up 175 units, so that with its space we have seven spacebands and a remainder of 340. The word "our" subtracts 110 plus 30 for the spaceband. Now, with eight spaces, we are reduced to 200 units. When we process "typesetting" we find that it totals 395 units and obviously won't fit. Therefore we back off, restore the remainder to 200 units and add back the value of the last spaceband, since if we do not include the word we do not need and do not want the space between "our" and "typesetting." We now have 230 units to account for. We check our spaceband tally, correct that count, and discover there are seven. We are allowed a spaceband expansion potential from 30 to 60. We try to justify the line by expanding the interword space to its allowed maximum. Seven spaces times 30 units equals 210 units, but we have to come up to 230. What can we do?

We restore the count to 230 and eight spacebands. We then go into the hyphenation routine. We bear in mind that if we have to hyphenate we have to add the hyphen, which, let us suppose, is 35 units wide in this particular type face. And so we have to find a word break within 195 units. We move over to the last letter that will fit within the line count and test for a hyphen point. We find that "types" equals 190 units and that this is the most that will fit, since "types" brings the total to 230. We test for a hyphen point after the "s" and hopefully we get a negative answer from our hyphenation routine. We back up and test for a hyphen point after the "e". We are told this is acceptable. Including the hyphen we have accounted for 185 points out of the available 195. The ten which remain constitute a surplus of units that our program knows can be distributed across the eight spacebands. And so, as we output the line, moving it to work area (C), where we shall store finished lines to be written to magnetic tape or punched out on paper tape, we indicate, somehow, that the values of the interword spaces shall be as follows: 32, 32, 31, 31, 31, 31, 31, 31. (This assumes that our photo-unit can cope with the precise increments of space as indicated.)

It may be seen, then, that the logic of justification is not difficult. For the moment we are regarding the hyphenation routine as a "black box" that is plugged into the system. In other words, it is completely independent of the logic of justification. Justification involves establishing a line count, reading in characters, keeping track of spacebands, decrementing the line counter, and determining when the counter goes negative. When this happens you back up and try to space out the line if this can be done. If it cannot be done, and the hyphenation routine provides a negative answer, all existing programs that we are familiar with can offer no solution better than to

insert extra space (beyond the maximum limits defined) between the words in the line. This may not produce bad results. In our example, if we could not have found a hyphenation point for "typesetting" we would have had to go to six spacebands of 63 units each and one of 62 units. The worst result would have been only 3/80ths of an em beyond the allowable interword space. The maximum would then have been 63/80ths of an em.

However, in other situations the interword space might prove to be excessive, especially where tight lines occur on either side of a loose line. When a linotype operator encounters this condition he may back up and try to reset the preceding line or lines to get a better distribution of white space. This procedure is, of course, highly desirable, and there is no reason why the same things cannot be done by computer program. However, a more complex logic pattern is required. The present writer describes this method as a "moving average" concept of justification, since the object is to try to average out the flow of white space over all of the spaces in adjacent lines of running text.

### Set Width Tables Are Vital

The vital role of set width tables should be apparent at once. Just as the justifying or counting keyboard has to access character widths so that the operator can make his own end-of-line decision, so must the computer know the relevant width values. These must somehow be available to the computer, either because they reside in core or can be brought into core when they are needed.

If core is restricted there may not be room to contain all of the tables that will be required for a particular job. It may well be that many such tables are necessary. One factor that will influence the number of such tables is the manner of storing width values. If each point size has to be described by a separate table then much room in core may be necessary. It seems better, therefore, to store *one* table for each font within a given size range, and to develop a formula to take into account the point size or set size of the type called for in a particular application.

Some typesetting machines today can create type in half point increments. In the design of our system we should therefore consider the possibility that we may want to generate type in 7½ or 8½ point set sizes. It seems desirable moreover, to express our unit values in terms which permit us to reduce them to a common base without incurring rounding errors. In doing so we must also be able to relate to the manner in which width values are expressed by the type designer or within the phototypesetting system(s) that we wish to access. The common denominator, in a relative unit system, could be 18 units, or 32, or even 80 per em.

The problem may thus be posed in these terms. We are working with a type face designed for a 32 unit system. Suppose we wish to mix sizes in a line. We start out with ten point type and encounter a character defined as 21 units, so that we now have 21/32nds of a 10 point em. Now we want to set the same character, in the same line (perhaps as a superior) in five point type. We must express the width as 21/32nds of a five point em. Finally, we want to set the same character, in the same line, in nine and one-half set, and we must describe it as 21/32nds of a 9½ point em. How do we decrement these values without incurring any rounding errors? There are ingenious solutions to this problem, but they are not well known. In fact, the problem is little understood.

To return to the basic dilemma of width storage, if we can contrive to have only one

table per range size per font, we will still be confronted with the need for many tables. If they are all to be stored in core simultaneously there is less core left over for other uses. One solution is to bring in the width table for the new font only when it is called for. This implies that the computer must have access to additional storage. Perhaps there is a disk or drum as a peripheral and width tables can be brought in from this source. Or perhaps they can be retrieved from a program library magnetic tape. Worst of all, the computer could stop and call for the operator manually to load in a new deck of cards.

The best solution is to be able to contain most tables in core and to have supplementary ones on disk. But the system can encounter substantial delays if it is continually engaged in overlaying width tables, despite high rates of information transfer. On the other hand, if all tables required are kept in core, and ten different fonts need to be accessed, (and if each font consists of 100 characters, each of whose width values can be described by one byte) at least 1,000 bytes must be reserved for this purpose. Experience of one user suggests that more than fifteen sets of width tables in core is highly desirable, although this user does not have a disk-oriented system. The throughput speed of any system, on complex work, will be profound influenced by the length of time it takes to obtain character width information. This of course must be done every time a single character is processed into the system.

### The Underlying Justification Concept

If we think of the justification program merely as one that is designed to accept a stream of text and arrange it into lines we will not do justice to the task that confronts us. Perhaps we moved too early into a discussion of justification, without sufficient attention to the fact that the underlying task is really one of *composition*, in which justification is only one aspect.

Even if we conceive of page makeup as being an essential element within our program package, and assume that the pagination program will be able to cope with the difficult assignment of achieving a proper positioning of lines vertically down the page, we still may be guilty of an imperfect conception of the nature of the problem. The fact is that *composition* goes far beyond the mere arrangement of text into justified lines and justified lines into made-up pages. It really has to do with the aesthetic treatment of consecutive areas of white space upon which typographic and graphic images are to be superimposed. If we think of the task in this light our entire approach may well be different. We may be more concerned with the arrangement of *areas*, in terms of x-y coordinates, than with the formation of lines of text. Especially for a typesetting device like the Linotron 1010, which has the capability of producing images at any point on a two dimensional surface of a certain size, an area composition concept will be helpful.

One set of programs available for typesetting (the RCA Page-1 or Page-2 programs) in fact starts out with this premise, and it conceptualizes a set of "boundaries" which coincide with the dimensions of a page. The first task of the makeup man or job planner (he is sometimes incorrectly called a programmer) is to define these boundaries.

We have not yet progressed to the point where we can truly think of composition in these terms. Someday in the near future we will be able to sketch a page outline on the face of a CRT display tube, and then "pour" the text into the area so indicated and study the results to determine whether or not the effect is pleasing. And so the

conceptualization of typesetting with reference to the length of the measure is perhaps a poor way to approach the arrangement of text in the composition process. But we are still bound by the traditions of conventional typesetting. All existing typesetting machines do have left margins which serve as points of origin (unless the turnpage feature of some tube typesetters is utilized, in which case there need be no left margin at all—only a beginning.) As a practical matter, however, most composition of today is concerned primarily with the *content* of the text rather than its *form*, and the emphasis upon line justification is consistent with this interest. (Advertising composition is the least content-oriented, with periodicals coming next. Newspaper setting is probably the application with the most conventional treatment of text.)

Why should we turn at this point to a discourse on the inhibiting aspects of the justification concept? Our reason is that the so-called hyphenation and justification program needs to have substantially greater capability than "mere" hyphenation and justification and it is best, at the very outset, not to lose sight of other system requirements.

### Tasks to be Performed

What are some of these tasks? The first and most obvious is the *quad left* function. Lines that are quadded left are not in fact justified (although they may be under certain circumstances). There are two situations to be considered with respect to the use of the quad left. One is its use to signal a paragraph ending, where the beginning of the line is not known in advance. The other is its use to signal a line ending where the line beginning is known and it is assured that all of the material from the beginning to the end will in fact fit on the line. This latter case is rarer than one would think, since one of the advantages of the computer method of composition is the ability to redefine line lengths and point sizes, thus permitting more or less text to fit into the line.

And yet the treatment of these two conditions needs to be distinguished in some way. For example, when we are dealing with the use of the quadleft code to signal a paragraph ending we can stipulate these conditions to govern its use:

1. if the line almost fits its measure, space it out;
2. if the line falls short by a defined amount (how defined?) use "normal" (not minimum) interword spacing;
3. if the line is very short, consisting of only a word or a portion of a word, consider the possibility of reprocessing the latter portion of the paragraph in order to pull it up.

Presumably, none of these conditions would apply to the self-contained quadded line. This latter case is apt to be a head or portion thereof:

BOROUGH COUNCIL REINSTATES WEST
CHESTER POLICE CHIEF

This suggests that West was reinstated as the Chester Police Chief. What the head intends is:

BOROUGH COUNCIL REINSTATES
WEST CHESTER POLICE CHIEF

Therefore we may need a different signal when we wish to quadleft self-contained lines, or if we use the same signal, placing it after "reinstates" and "chief," we would not wish to activate the subroutines for spacing described above.

However it is not always possible to determine in advance whether certain material will fit on a given line length, even if we know the measure and that it will never be changed. This fact highlights one of the great disadvantages of computer composition. A linecasting machine operator can see whether words will fit and how they will look. A TTS operator, a Monotype keyboard operator, or anyone producing input on a counting keyboard, gets an immediate feedback. The off-line keyboard operator, producing unjustified input for a computer composition system gets no such feedback. Perhaps the markup man can count out width values, or estimate them in some fashion, but this is laborious and inefficient. It is probably better to wait and see how the computer handles the problem and then effect changes. Or, perhaps a sophisticated program, assisted by input coding, could deal with the options.

The problem of the self-contained quadded line comes to light not only in the setting of heads and other display material, but also in the setting of poetry, or other types of lines, such as listings, that may or may not "turnover." We therefore find it necessary to define, in the event the line does turn over, the conditions we wish to govern. Should hyphenation be permitted? Should the second line be indented, and if so, by how much? Are there places where the first line should not break, even without hyphenation? Should the first line be left ragged or be justified?

I look to Thee in every need,
And never look in vain;
I feel Thy strong and tender love,
And all is well again.

This is certainly better than:

I look to Thee in every
need,
And never look in vain;
I feel Thy strong and tender
Love
And all is well again.

or, perhaps even worse:

I look to Thee in ev-
ery need
And never look in vain;
I feel Thy strong and ten-
der Love
And all is well again.

Some of these problems will never be solved economically in computerized composition until we get immediate feedback from the system which permits the input operator to appraise the aesthetic consequences of his input or permits someone else easily to improve upon them. We could, however, program an optional break code: "if you can't get to the quad, break here," it says.

### Quad Right and Center

The "justification" program must also be able to quad right a self-contained line. This involves inserting "normal" rather than minimum space between the words in the line, and positioning all extra space (which must be calculated) to the left of the first word. Presumably, however, if the information will not fit on the line when "normal"

or average interword space is used, it would be desirable to go to a minimum interword space before letting the line break over. If the line does not fit, then somehow a judgment should be made as to whether the first line should be justified (with hyphenation?) or set flush right, and whether the rest of the material, which runs over, should also be flushed to the right, or treated in some other manner.

How Should You Treat Quadded Matter that Breaks Over?

The same problems are posed with respect to the use of the quad center code. Here the program must take the extra space and distribute it equally on each side of the line. If it will not fit within the measure, should it attempt to justify the first line?

How Should You Treat Quadded
Matter That Breaks Over?

How Should You Treat Quadded
Matter That Breaks Over?

### Flush Left and Right

Another condition that requires set width calculations in order to handle leftover space is the setting of material both flush left and flush right, with all the remaining space in the center.

Flush Left Flush Right

And here again, since w/ are dealing with general conditions, we must define what should happen if the elements do not fit within the line:

Flush Left for this Portion Flush Right
for this portion

Flush Left for this Portion Flush Right
for this Portion

Flush Left for this Portion Flush Right
for this Portion

### Leadering

The same problems exist when it comes to leadering. The program should offer the capability of filling in the extra space with whatever characters we might wish to specify:

Flush Left.....Flush Right

Flush Left . . . . . Flush Right

Flush Left ----------Flush Right

Smith, John.....922-4567

American Society for the Advancement
of Barber Shop Quartet Singing,
Philadelphia Chapter.....922-4568

And horizontal rule generation is a function of the same program, although it can be handled by other commands as well.

x _____ x

### Changes of Measure and the Use of Indents

The program has to be able to cope with instructions to change line length which are embedded in the text. They may or may not occur at the end of a paragraph. In this case we have given a command to change line length from 36 picas to 24 picas at the beginning of the next line after the command is given rather than at the beginning of a paragraph. But we could have accomplished the same result by a command to insert a right indent at the beginning of the next line. If the photo unit sets from left to right a shortening of the line length accomplishes the same result, in galley material, as a right indent. But if the photo unit sets from right to left, which is a peculiarity of the Photon 513, then a shortening of line length would be equivalent to specifying a left indention.

Indentions are of various types. There is the indention of specified duration, which is to continue for a given number of lines or points and then is to be dropped. Or there is the indention of indefinite duration which is to be continued until a signal is given that it should be discontinued. There must be a capability to handle left, right or center indents, and to create hanging indents, and to calculate the amount of indention required to align with a string of characters whose total width the input operator does not know. Moreover, there must be a capability to nest indents. Here are examples of some of these conditions.

Within our overall line measure of 36 picas we have now called for a center indention of 10 ems and we have specified that the material we are setting is to be hyphenated and justified within this center indent.

Now we have called for a left indent of two ems additionally, to be nested within the center indent and we shall now specify that a right indent of six ems shall start on the next line and shall continue for a total of six lines, at which time it shall automatically terminate. It may thus be seen that we have nested a right indent within a left indent within a center indent. We shall permit the right indent to cancel itself and then shall insert codes for the cancellation of the left indent. When we return to the center indent we shall

proceed to illustrate the hanging indent condition. The hanging indent is used for lists and outline material.

This is an illustration of a hanging indent of two ems which is to continue until the program encounters the end of paragraph quad code.

The indent will automatically resume for the second line in the next paragraph and this condition will remain in effect until it is cancelled. Now we shall cancel it and illustrate a variation of the hanging indent, called a variable indent, which automatically calculates the amount of hanging indent to apply in order to achieve certain alignment conditions.

1. Our task is to achieve alignment with a set of figures, so that turnover lines do not obscure the numbers.

10. The reader will notice that we have also aligned on the periods, by keying a nut space to the left of the single digit numeral.

**Ragged vs. Justified Setting**

The program should also have the capability of setting material ragged right, ragged center or ragged left, with or without hyphenation. Here are examples of the various effects which can be achieved by the use of these commands. In order to enable the reader to make comparisons we shall set the paragraph in different ways.

First, aligned on the left, ragged on the right, without hyphenation:

Here is a sample paragraph we shall set several ways in order to illustrate the difference between hyphenated and unhyphenated material, and ragged setting versus justified setting. We are setting all of this paragraph in a center indent.

Now, still aligned on the left, ragged on the right, but with hyphenation:

Here is a sample paragraph we shall set several ways in order to illustrate the difference between hyphenated and unhyphenated material, and ragged setting versus justified setting. We are setting all of this paragraph in a center indent.

Now, aligned on the right, ragged on the left, without hyphenation:

Here is a sample paragraph we shall set
several ways in order to illustrate the
difference between hyphenated and
unhyphenated material, and ragged
setting versus justified setting. We are
setting all of this paragraph in a center
indent.

Now, aligned on the right, ragged on the left, with hyphenation:

Here is a sample paragraph we shall set
several ways in order to illustrate the
difference between hyphenated and un-
hyphenated material, and ragged setting
versus justified setting. We are setting
all of this paragraph in a center indent.

Now, "ragged center" without hyphenation:

Here is a sample paragraph we shall set
several ways in order to illustrate the
difference between hyphenated and
unhyphenated material, and ragged
setting versus justified setting. We are
setting all of this paragraph in a center
indent.

Now, "ragged center" with hyphenation:

Here is a sample paragraph we shall set
several ways in order to illustrate the
difference between hyphenated and un-
hyphenated material, and ragged setting
versus justified setting. We are setting
all of this paragraph in a center indent.

### How to Achieve These Effects

It is obvious that codes must be given to the computer to instruct it to change its mode in order to reflect changes in line length, justification, hyphenation, ragged setting, or to set up indention conditions. It is sometimes possible to ask the program to infer these specific conditions from the identification of a general typographic situation. The difference between the general set of conditions and the specific definitions will be considered in our chapter dealing with the use of "Macros" and format statements. At this point let us assume that all instructions are specific. We have already seen that we can convey a line length instruction by the use of a code *M3600, for example. We have also seen how we can define one type face as *R and another as *I. If we wish to save all other letters of the alphabet for different type face designations we must come up with some other instruction configuration for command codes to implement some of the things we have been discussing in the latter part of this chapter. These codes could be mnemonic, with their functions suggested by their letter combinations, or they could simply be numeric. For example, we could let *00

indicate justification and hyphenation, and *01 indicate justification without hyphenation, while *20 would signify ragged-on-the-right with hyphenation, and *21 ragged on the right without hyphenation. All codes beginning with "9" could relate to indents, and the amount of indention, in ems, could follow a letter which defines the kind of indent. Hence *9c06 could mean a center indent of six ems.

The command structure of a typesetting program is often imbedded in the text. It can be devised to be understood by amateurs or only by professionals. Some such systems today are highly complex and difficult to work with, and they may be quite sophisticated. Others are easier to understand but difficult to keyboard. For example, one common approach uses "delimitors" to separate commands from the text stream. These delimitors are represented on the hard copy of the input device as left and right square brackets. [tf 15;ps8;bll0] would describe an instruction to justify according to type face number 15, set in eight point type with 10 point body leading. [ns, 1] means to end the paragraph, start a new line and skip one line in so doing.

Unless all of the typographic elements can be inferred by the text structure, which is rare, the input operator, perhaps following instructions from the markup man, must assist the computer program by giving it the commands it requires in order to know how to process the text material. These codes may be minimal or extensive, depending upon the complexity of the typography and the demands of the program.

**Review Questions**

1. Explain how the computer program justifies the line.
2. Why are tables of width values important to this process?
3. Why define a minimum and a maximum value for an interword space?
4. Why distinguish between a fixed space and a variable space?
5. Why are separate width tables required for different type fonts or faces?
6. Would separate width tables be required for different point sizes of the same type face? Why or why not?
7. Under what circumstances would the program call in the hyphenation routine?
8. What does the program do if it cannot achieve hyphenation?
9. What is meant by a "moving average" solution to the justification problem?
10. What problems are posed by changing point size or set size in a given line?
11. Can you define any typographic situations which have not been dealt with in this chapter that you would want a composition program to be able to cope with?
12. How would you handle the setting of tabular matter?
13. How would it be possible to incorporate a feature which would underscore certain words or sentences?
14. How could you generate vertical rules?
15. How could you set multi-level mathematics?
16. How can the computer program "know" how to distinguish between text stream information and typographic codes and commands?

CHAPTER XII

# Hyphenation Examined

A hyphenation routine is an essential part of a justification program. Without hyphenation the quality of justification is unacceptable except perhaps in very wide measure composition. There are techniques for "hyphenless justification," but without some recourse to letterspacing the results are such that one might better consider ragged setting as a more pleasing alternative. (Even in ragged setting some hyphenation is desirable to conserve space and to provide for a more pleasing appearance.) Here is an example, in narrow measure, of hyphenless justification. However, the program which composed this sample contains an arbitrary limitation on the amount of interword space it can assign. Where the interword space required exceeds this maximum the line may not be justified under any circumstances.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

Now let us set the same material *with* hyphenation.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

It seems clear enough that if margins are to be justified, hyphenation is an essential exercise in order to avoid awkward and unseemly interword spacing. To the extent, then, that valid hyphenation points cannot be discovered by the program, and no hyphenation occurs, the quality of justification suffers.

It is also evident that the frequency of hyphenation, within a given text element, will be a function, among other things, of the amount of interword spaceband expansion which is specified. Here are examples of the same text set with different parameters for interword spacing, resulting in more or less frequent hyphenation, and some impact upon the quality of justification.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

But the fact that the program calls for hyphenation does not assure that the words submitted to the hyphenation routine will in fact be divided. These are words which are encountered at the "fall-off" point—that is, where their inclusion is impossible, but their exclusion would necessitate overexpansion of the spacebands beyond the amount stipulated. Clearly, not all the words which are so encountered should or could be hyphenated.

### What Words Should Be Divided?

The first decision to make, therefore, is whether to submit the fall-off word to the hyphenation routine in any case. In this connection it is almost safe to say that accepted practice would rule out the hyphenation of any word of less than five "alpha" characters. (The term "alpha" is used to emphasize that other symbols may be attached to the word—notably punctuation marks, but perhaps other characters as well.) This "rule of five" arises from the injunction that, generally speaking, not less than two characters should be left behind and not less than three should be carried over. Thus, not all five letter words of two or more syllables should be divided: a/gain, a/head and may/be come to mind as examples in point.

### What Constitutes a Word?

Another question to be asked is what is a *word* for purposes of hyphenation? The first step in the hyphenation routine must consist of some procedure to "clean up" the

word for analysis. Then, after finding one or more hyphenation points, the original or "dirty" word is reinstated.

For example, if the computer uses a six-bit configuration, upper and lower case alpha characters will probably be differentiated by the presence or absence of precedence codes. Presumably these indications would be removed during the "clean word" routine, and yet capitalization may give a useful clue to word division, as in the case of McLean or MacIntyre. Another type of problem associated with the "cleaning up" process may be illustrated by the treatment of "and/or." If the slash is removed during the clean word routine, since it is a non-alpha character, the word appears as "andor." If the hyphenation routine then determines that the proper division of the word is *an-dor*, as it probably would, then the "dirty word" routine will put the slash back, so that the result will be *an-d/or*. Perhaps the only solution to this problem is to regard the slash as a word-break character, like a space (except that the space is dropped when it falls at the end of a line, whereas the slash would be set). Since a slash may also be used in the composition of fractions, the word-break routine will have to distinguish between the use of the slash in the context of alpha characters and the presence of a slash surrounded by numerics.

Here are illustrations of other types of situations that may arise:

| | |
|---|---|
| computer." | perhaps* |
| "However, | $special |
| hurry! | computer[a] |
| forever? | mis*taken* |
| maybe—but | mañana |
| illustration." [1] | *καλον* |

It would be possible to write an entire treatise on the subject of how to determine which "words" to submit to the hyphenation routine. All that we can do in this presentation is to identify the problem.

### The Vowel Group Analysis

It shall now be assumed that we are presenting a clean word of five or more letters to the hyphenation routine. The next task is to determine whether or not the word can be hyphenated. By definition, only words containing more than one vowel group can be divided. Words such as "through" or "straight" obviously cannot. A routine is needed, therefore, to identify vowels and consonants and check to see if there are two or more vowel groups (vowels separated by consonants). If we represent a vowel as "0" and a consonant as "1," the word *computer* translates to 10110101. Scanning across this string establishes that there is more than one vowel group. Still there are problems. One such problem is to determine whether "y" is a consonant or a vowel, as in "dystrophy" as opposed to "leapyear." Another is to distinguish between the "silent" *e* as in *moose* or *smoothed* and a non-silent *e* as in *ragged* or *rages*.

Perhaps no such refinement is required. Perhaps this problem can be dealt with in the string analysis of alpha characters, so that at this point our only task is to determine that the word *may* be hyphenable. However, at some place in the routine this difficulty will be encountered and must be resolved.

**What Hyphenation Programs Do**

In general, hyphenation programs consist of one or both of two kinds of procedures. One is a "logical" approach. The other is a dictionary "look-up." Let us consider the *logical* solution first. As a rule such algorithms do not examine the entire word (at least at one time) but only inspect a portion of it. In fact, one of the most common algorithms tests to see if a hyphen may be placed between the second and third characters of a four-letter string. If the word which straddles the fall-off point is "photographer," and the last two letters that could be included within the line are "gr," then there would appear to be no point in testing more than two letters beyond this location. Hence, the first interrogation would be made of the tetrad:

photo(gr/ap)her

Three questions might be asked:

1. What are the possibilities of a hyphen occurring *before* "ap"? The answer might be that the chances are relatively high.
2. What are the possibilities of a hyphen occurring *between* "r" and "a"? Again, the possibilities might be good.
3. What are the possibilities of a hyphen occurring *after* "gr"? The answer would most certainly be negative.

The logic program, being based upon an analysis of the English language, should be able to ascertain that no syllable can end with a "gr." Hence, this hyphenation point would be rejected, even if the other two queries were favorable.

The program would then shift one character to the left, and look at

phot(og/ra)pher

Here, presumably, we should get an affirmative answer to all three questions. This hyphenation point would be accepted. Note that the program would never look at the "whole" word. But even if it did, it would confine its attention only to a given four-letter combination at one point of time. Hence, though it would achieve a correct answer for *photographer* it would get a wrong answer for *photographic*. However, if only "photo" would fit, so that the first tetrad examined would be "to/gr," then a "yes" answer would be produced, since hyphens do occur after *to*, between *o* and *g* and before *gr*. This answer would be correct for *photographic* but would be wrong for *photographer*, indicating that the right or wrong solution in the hyphenation of this word is a matter of chance.

Another approach would be to start from the beginning of the word, rather than with the tetrad which includes the two characters beyond the fall-off point. The trouble with this procedure is that it presumably would accept the first solution encountered even if more characters could be squeezed into the line. It appears that moving from right to left will produce tighter lines than moving from left to right.

Still another alternative would be to continue across the entire word, in either direction, logging all of the acceptable points and assigning a probability value to them. Then the best solution would be selected (if it fits) regardless of whether it was the tightest or the loosest. The program might come up with such statistics as the following:

| *tetrad* | *probability* |
|---|---|
| ph/er | 90 |
| ap/he | 70 |
| ra/ph | 85 |
| gr/ap | 0 |
| og/ra | 80 |
| to/gr | 85 |
| ot/og | 60 |
| ho/to | 80 |
| ph/ot | 60 |

In this case the first three solutions would be discarded because they would not fit in the line. The highest probability solution remaining would be to/gr (which, unfortunately, would be wrong!).

Our illustration has already prejudiced our analysis of this type of hyphenation routine, but we chose a difficult example! Whether you inspect the whole word or only a portion of it, if the answer is found by looking only at a portion of the word there is no assurance, other than coincidence, that the right answer will be achieved. Of course, there are certain sure-fire combinations, but these, unfortunately, are relatively rare. For example, one can be fairly certain that:

| *a syllable will not end with* | *a syllable will not begin with* |
|---|---|
| bc, bd, bg, bh, bj, bk, bm, bn, bp, bq, br, bv, bw, bx | bb, bc, bf, bg, bh, bk, bm, bn, bq, bs, bt, bv |

If we take just these cases of consonants beginning with *b* (and our list should be extended to consider *all* consonants) we find hyphen points after the first category and before the second would be prohibited. Notice that this particular test is useful (if our empirical observations are correct) not for the purpose of finding a hyphen point, but for the purpose of rejecting one once found by other techniques. One program we know of makes all of its hyphenation judgments solely by an analysis of vowels and consonants. Once such a determination is made, a second check of the kind indicated here would certainly be helpful!

It is not, in fact, very difficult to find acceptable syllable breaks by studying combinations of two or more consonants. There are, of course, certain problem combinations. Where two *b*'s occur together the chances are greater that the hyphen would appear *between* them, but it could occur *after.* Where *bbt* occurs we can be certain that the hyphen would follow the second *b,* as in Crabbtree; but where *sth* occurs the word could be *masthead, asthenia* or *asthma.* The *st* combination is particularly difficult, since we have as/ter, cast/er and be/stir.

Hyphenation procedures are much more complicated in those cases where we are dealing with a single consonant with a vowel on either side. For example, consider bas/il and ba/silic, ba/sin and bas/inet. On the whole there is probably only a 50% chance of being correct when dealing with an otherwise unidentified portion of a word containing a single consonant at the proposed hyphenation point, such as

| | |
|---|---|
| re/si | re/side, res/idence |
| re/se | re/serve, pres/ence |
| em/or | fem/oral, de/moralize |

The accuracy of the algorithm may be enhanced somewhat by taking a five- or six-letter combination, such as:

de/mor or dem/ora

but it is doubtful whether the inclusion of one or two additional characters in the diagnostic string will in fact contribute a great deal to the solution of the problem.

### Word Structure Analysis

Another approach to an algorithm involves a more substantive analysis of the structure of the word itself, aimed at identifying prefixes and suffixes and root words, and treating them differently. As a general rule, for example, it is permissible to hyphenate after a prefix such as *re* in *reentry* or *restructure.* Another example might be *dis* or *mis, in* or *un,* or perhaps *trans.* Suffixes such as *er, tion, sion, cious, tious, ceous, able, ible, ness, ish, ate, ive,* and *some* may likewise be acceptable post-hyphen points in many cases.

Looking first at the prefix possibilities, we may generalize that a hyphen may almost always be placed after *in* at the beginning of the word, but there are a few exceptions, such as *inchworm, inkless* and *innkeeper.* The prefix *un* would give us trouble in *union, unction, unicorn, unified, unison, united* and *universe.* The prefix *dis* wouldn't work for *dishwasher,* and *mis* would cause difficulties with *misogyny* and *mistiness. De* would bring trouble in the case of *decalogue, debonair, debit, debutante, decadence* and many others. *Anti* as a prefix would produce the wrong result for *antiphony* and *antiquity. Re* would be most unsuccessful. Witness *reappear* and *reaper, react* and *ready, recognizance* and *recognize. Inter* would present problems for *internship* or *interior.*

Looking at suffixes, there is dream/er and boul/der, check/er and chat/ter, chas/er and Chau/cer, control/ler and poll/er, floun/der and bound/er. Or consider ca/pable and wash/able, commu/nism and social/ism, patron/age and out/rage, medi/cable and limit/able, contempt/ible and admis/sible, consum/mate and elector/ate, ad-mon/ish and bran/dish. Clearly, there are relatively few trouble-free suffixes. Probably *tion* and *some* and *ness* are the most consistent. The letters *ing* are unusually difficult, for we have such peculiarities as let/ting and grat/ing, grasp/ing and shop/ping, stall/ing and appal/ling, whis/tling, whisper/ing and confer/ring.

Of course, many of the apparent contradictions are capable of resolution. An *ive* generally seems to be preceded by a consonant which is affixed to it (*res/tive, mo/tive,* even *pas/sive*) but logical algorithms to differentiate the rules to handle apparent inconsistencies may become very involved. The plain fact is that the English language permits the division of words in singularly arbitrary ways and the rules are not always susceptible to analysis on the basis of the facts at hand, since they have something to do with the pronunciation of particular vowels, the derivation of the word, or its part of speech. The proofreader might know the rule that the *ing* stands by itself when it follows a double consonant that exists in the root word *(pass/ing)* but that there is a division between double identical consonants when the root word does not contain both letters and the second is added as an aid to pronunciation *(let/ting).*

There are still other aberrations. There are differences between British word division and American. (*Pro/gress* and *prog/ress* for the noun; *pro/cess* and *proc/ess* for the verb). The British pay more attention to the derivation of the word *(demo/cracy* instead of *democ/racy, know/ledge* instead of *knowl/edge).* And even in the United States more emphasis upon word structure and less upon pronunciation is reflected in the word

division principles of "Webster's Second" as compared with "Webster's Third" dictionary. In the latter, more and more popular as a guide in the United States, pronunciation tends to take over, so that the word becomes *concor/dant* instead of *concord/ant,* and *depen/dence* instead of *depend/ence.*

Difficult as it is to program good word division, at least it does not present the complications of Swedish, where the rule is that there are certain letters which must be repeated when a word is hyphenated. And so *glasstrut* (ice cream) becomes *glass/strut* when it must be split! (A similar condition appears in the German language.)

But our description of the problem is not yet over. The rules draw a distinction between syllabification and hyphenation. (The reader is cautioned that many dictionaries do not!) The Chicago Manual of Style specifies, for example, that word division should be made after a vowel, "unless, by so doing, the division is not according to pronunciation. Where a vowel alone forms a syllable in the middle of a word, run it into the first line." [1] Hence *commemo/rate* is better than *commem/orate,* and *physi/cal* is to be preferred to *phys/ical.* On the other hand, *ac/etate* is probably better than *ace/-tate,* and *Bab/ylonia* might be preferred to *Baby/lonia,* although neither is a very happy solution.

Where there are two vowels that are pronounced as separate syllables they should be split, as *initi/ative* rather than *initia/tive.* Furthermore, compound words should be split, where possible, so that their elements are kept together. *Ante/cedent* is preferred to *an/tecedent,* and *homo/geneous* to *ho/mogeneous.* Where words are joined by hyphens already, one should avoid placing another hyphen, if at all possible *(quasi-public,* not *quasi-pub/lic)*

One wonders, therefore, whether *any* algorithm can do justice to the problem of hyphenation. Moreover, there are the homographs, which generally hyphenate differently when they have different meanings. A list of these, prepared by Howard Bezanson of Waverly Press, appeared in the *CIS Newsletter* for October 1, 1970, and they are worthy of capturing here in a more permanent form. Consequently, they are reproduced herein:

> acerous, adept, agape, anchorite, arsenic, associate, bunter, coster, crater, decameter, denier, desert, divers, dosser, draper, ellipses, ergotism, ergotize, evening, former, founder, fuller, gaffer, fainer, gaiter, genet, genial, gilder, graining, groper, grouser, halter, hinder, homer, hurter, impugnable, invalid, leaguer, legate, limber, luster, master, minute, nestling, petit, periodic, pinky, planer, prayer, precedent, present, project, produce, product, progress, putter, putting, rafter, rebel, record, refuse, resume, rosier, spiller, stater, stingy, stover, tamper, tenter, and welter.

To this list of 70, we add baton, Arundel, recreate, decanel and Eunice. In addition, there are many proper names that may be confused with other nouns, such as *Weller, Baker, Miller* and *Wheeling.*

Obviously nothing can be done about these homographs, since a contextual analysis would be too elaborate for a production typesetting program. But one solution might be to "flag" them so that the proofreader will be able to locate them and rearrange the hyphens if the wrong sense of the word is elected.

[1] University of Chicago Manual of Style, 12th ed. University of Chicago Press, p. 137.

**The Exception Word Dictionary**

In view of all of the problems, it is not possible for a logical program to yield good hyphenation results, if our criteria for *good* results take *frequency, accuracy, throughput speed* and *core requirements* into account. The safest thing to do about a combination of letters which breaks differently under different circumstances is not to hyphenate at all. If the program rejects all such conditions the frequency of hyphenation will be extremely low. While too frequent word division is undesirable (such as three or more lines in a row) the occurrence of hyphens should be determined by parameters and not by the exercise of caution in a cowardly hyphenation routine. The program should be good enough to find most of the valid hyphenation points in almost every hyphenable word. If it fails to do so, at some point of time poor justification will result.

Accuracy is important not only because a bad hyphenation point detracts from the authentic nature of the product, but because to correct it requires the resetting of *two lines.*

Throughput speed and core requirements are factors which profoundly affect the cost of the typesetting system, and the former may be of importance where an improvement in turnaround time provides one of the justifications for the installation of a computerized typesetting system.

One reaches a point, therefore, where further improvements in the logical aspect of the program, in the interest of more frequency and better accuracy, make great demands upon throughput and core, and it therefore becomes more expedient to try to deal with the complications of hyphenation by using an exception word dictionary to supplement the logical program.

When such a dictionary is used, the word which straddles the fall-off point is first sought in the stored dictionary. If it can be found there the hyphen points are given. If it cannot be found it is assumed that the word may be divided reasonably according to the logical rules of the program.

One of the earliest routines for hyphenation was developed by Perry Publications, in West Palm Beach, in late 1963. A large dictionary was stored in an optimized fashion on a magnetic tape which could be searched by "read forward" or "read reverse" instructions. The words on this dictionary were those which could not be broken after the third, fifth, seventh or ninth character. *Sedentary* would not be found in this dictionary, nor would *secretary* or *boreal.* Some words that did not follow the 3-5-7 rule might also not be considered as exceptions if some other criterion kept the routine from doing any harm. For example, *cactus* divides after the third letter, and while it does not divide after the fifth, the routine that requires two or three letters to be carried over would prevent a hyphen except after the third letter.

Since there are many exceptions, this simple logical program needed a big dictionary. If the logic is more elaborate the dictionary can be smaller, but one unfortunate implication of an exception word dictionary is that it has to be searched before the logical rules can be implemented. (Another limitation is the assumption that if the word is not specified as an exception, it follows the rule. The fact is that the word simply may not have been included, either in ignorance or for lack of room.)

Exception words are usually stored in core rather than on tape or disk, because core can be accessed more quickly, and can be indexed, if necessary, for random access. Because of core limitations, not all exceptions can be stored. But whatever means is used to store the dictionary, access time is exceedingly important. Tape, which is

inevitably sequential according to some formula, is the slowest. Because a preliminary dictionary look-up is necessary for the determination of whether or not the logic rules may be applied, a combination of logic and dictionary look-up sometimes yields a routine which first explores letter groupings and then, if it encounters certain combinations where trouble is more likely, proceeds to the exception word search. For example, if the word break can be placed between *b* and *c* or before *tion*, there isn't much need to look up the word in question. Often only root words are stored (robbed of prefixes and suffixes) or certain letter conbinations. *Constellation* may first be stripped of the *con* and *tion*. If an appropriate break point cannot be found at either end of the word, or if the routine isn't certain it will yield a valid prefix, for example, of the sort where you can invariably break, then *stella* might be queried to see if the most reliable break point would be between the two *l*'s or after the second.

This writer's strong preference is for the treatment of *all* words as exceptions. If the system permits the use of disk storage of one million or more bytes solely for a dictionary, and if the disk has reasonably good access time, it is possible that correct *and preferred* break points can be found within the same time frame required to go through all of the logical interrogations that would otherwise be necessary. This writer happens, moreover, to believe that *preferred* hyphen points should be stored. In this case the results which can be achieved by the computer are likely to be more consistent and more satisfactory, from every standpoint, than the decisions arrived at on the spur of the moment by even the best compositor.

### Hyphenation Verification

However hyphenation is achieved, it is a useful adjunct to the routine to cull out all of the words which have been divided, to keep track of their position in the text, and to print out the solutions arrived at. This not only serves as a check on the accuracy of the solution (and enables the reader to locate the bad ones easily) but it can call attention to problems with the routine that may be correctible.

### Multi-Line Justification Solutions

It is possible to have the best hyphenation routine in the world and still get results inferior to those achievable by a human so long as the end of line decision is made in the vacuum of a single line of text. What makes the human operator's result better, in many cases, is his rejection of the line he has already composed. He then goes back and resets the *preceding* line, bringing a small word down or up, or dividing the fall-off word in a different place, to get better results on the line which follows. In the same fashion, we believe that a good computer routine should have the same capability, and should be able to "look at" five or six or more lines at the same time. If it can succeed in doing this, no human would be able to obtain comparable results in a consistent fashion.

And so, as we have seen, the hyphenation of text is indeed a complicated and fascinating subject! For scholars and programmers there will always be scope for further research and experimentation.

**Review Questions**

1. What is "hyphenless justification"? What does it imply in terms of the quality of justification?
2. Why does line length affect the frequency of hyphenation?
3. What relationship does the amount of allowable interword space expansion have to the frequency of hyphenation?
4. Does the kind of vocabulary in the text to be set (length of words used, as in scientific writing) have any bearing upon the problems of justification and hyphenation? Why?
5. What is meant by a "clean word"?
6. Why are vowel groups important?
7. What is a silent *e?*
8. What constitutes a "word" for the purpose of hyphenation? How would the hyphenation algorithm "know" whether or not the character string represents alpha characters?
9. What do we mean when we refer to the word that "straddles the fall-off point"?
10. What difference does it make if the program "looks at" the "whole" word as opposed to a portion of the word?
11. How is an "exception word" dictionary used in hyphenation?
12. Do you know any dictionary which distinguishes between syllabification and hyphenation?
13. Can you find any problems with the hyphenation of *acetify, bottleneck, record, cello, conciliate, copybook?*
14. How would a hyphenation routine deal with the word "coney"?
15. Can you give examples of preferred hyphen points?
16. How would you deal with the problem of three hyphens in a row?

CHAPTER XIII

# The Correction Program

A satisfactory method of correcting the "file" of information to be typeset is an essential element in any computer typesetting system which outputs to a photocomposing device. While hot metal linecasting can be corrected easily by the substitution of one line slug for another, cutting and stripping or pasting of substitute lines on galleys or pages of photo-composed type is by no means as easy to accomplish. It not only takes time to insert the corrections, but opaquing of the negative is usually necessary to get rid of the strip marks. Moreover, for any high-speed system which requires a substantial volume of typesetting to provide economic grounds for the installation, stripping in any quantity would be an anachronism.

We must accept the fundamental principle that photocomposition makes the greatest sense when two conditions can be met: input can be corrected so that output is clean, and output can be produced in fully made-up pages.

In our discussion of correction techniques, we shall ignore the possibility that corrections could perhaps be accomplished without the aid of a computer. There are ways of doing this, of course, through paper tape merging devices and the substitution of one punched card for another, but our interest is in the use of the computer to facilitate the use of the correction process, and this is what we propose to discuss in this chapter. The only exception will be a brief mention of making corrections through verification keyboarding.

We shall assume, therefore, that it will be possible to store the initial input in some medium so that it can be accessed for revision, and to derive a proof of this input, at the time of storage, so that the errors to be corrected can be detected. While it is possible to *reprocess* the *initial* input and apply corrections against it, the more common systems concept implies that the input will have been stored either on magnetic tape, disc or drum, and that it will have been reformatted for this purpose and for the purpose of making the correction process easier.

### What Do We Mean by "Corrections"?

We have deliberately used the concept of *correcting* rather than *editing* because we wish to draw a distinction between these two processes, even if there may be some conceptual overlapping. We shall discuss what we consider to be the true implications of *editing* in a later chapter, but we shall make occasional reference to the editing process in this present discussion. However, our primary emphasis will be upon a process which is somewhat more naive and simplistic. The tasks we shall be primarily concerned with have to do with verification—that is, to be certain that the information which will go into type will be precisely the same as the information set forth in the

initial manuscript, keystroke for keystroke, but that, in addition, the necessary command codes or flags for typesetting will somehow be included, and that these codes will also be correct.

### Verifying Procedures

**When we conceive of the correction process as one of verification, we are thinking more in conventional data processing terms, and we are thus led to consider conventional data processing verification techniques.**

**One such procedure consists of a double keyboarding of the initial input. This is often done by a keypunch operator who sits at a verifying punch, inserts the initial stack of Hollerith cards, picks up the initial manuscript, and proceeds to duplicate the keystrokes of the original keypunch operator. In the event that the same key is not depressed, the keyboard will lock, and the second operator will be able to ascertain whether it is she or her predecessor who is responsible for the mistake. Having made this determination, the correct key can be depressed and the new output, on a new punched card, will replace the old one.**

**A similar technique would be possible on a specially-devised paper tape keyboard, and it can also be done on a magnetic tape encoder such as a Mohawk.**

**While this may appear to be a relatively costly way to "proofread," the fact of the matter is that if it suffices it is probably the cheapest method of assuring that the input is accurate. However, it will probably not suffice for a final reading. Nevertheless, it is a technique that should be given serious consideration, especially where the material to be typeset is also destined for some form of data manipulation.**

We have said, however, that our interest lies in *computer* techniques. Consequently, this method of verification falls outside the definition of our assignment. On the other hand, dual input can be processed by a computer program and compared character-for-character. One of the two versions can be printed out, and deviations between that and a second version can be flagged. There are certain problems in doing this. In the event that there is a discrepancy between the two input streams the program must manage somehow to return to a common point of reference. However, this is not too difficult. The comparison can be resumed at the beginning of the next word space, the next period, or the next paragraph.

### Diagnostic Routines for Verification

Another type of verification may be designated as a *diagnostic* pass. While many computer programs contain special diagnostics to prevent the waste of computer time and to assist the operator or the programmer in discovering the source of his problems, special passes or routines are sometimes written to uncover common difficulties and to highlight certain conditions in the input stream. For example, if the program can sense the existence of a coded instruction which specifies the beginning of an indent, a parallel instruction should be present at some point in the input stream to signal that the indent condition has been cancelled. In like manner, an instruction to go into italics will normally be followed by a code returning the text to the roman face. If the diagnostic pass indicates that italic begins on line 22 and roman resumes

on line 300, the chances are high that a code has been dropped somewhere. It is easy enough to find out whether or not this is the case. Diagnostics of this character are useful in checking for typesetting codes. They are inconspicuous and are usually buried in the text stream. Yet the consequences which follow from the presence of one faulty keystroke of this nature are much more significant than those which flow from a normal typographic error. The preceding section of material in bold, illustrates our point very well.

### What Should be Read?

One of the most significant decisions in the definition of a typesetting system is to determine *what* it is that should be read in order to discover the inevitable input errors. Discarding the hard copy itself as inadequate for the purpose of proofreading (although it may serve other useful functions) the alternatives seem to be:

1. a computer printout;
2. an image generated on a CRT display screen of a so-called *editing* terminal;
3. typeset galleys or pages.

There are sound arguments for each of these approaches. We shall discover that each medium serves a particular function and has its own advantages and shortcomings.

### The Computer Printout

The printout may be obtained as a byproduct of another computer pass (assuming that the printout device is part of the system), although the chances are that the printer will run more slowly than other portions of the program and may cause the throughput to be "print-bound." Alternately, the printout may be obtained as a separate computer pass, or from an off-line printer.

Such printouts may be in machine coding, which makes them virtually unintelligible, especially if this coding is hexadecimal (although such a printout is often useful for program diagnostics). More likely, the printout will represent the text in an intelligible typewriter-like face, with certain conventions used to describe characters which are not normally a part of the typewriter character repertoire. For example, a *lozenge* may designate an "em" space, a *plus* sign, a "nut" space, and an *at* sign a "thin" space, to distinguish these fixed space elements from the justifying spaces which will be blanks. An asterisk could represent the function code flag, as we have indicated elsewhere. Three other special symbols might be selected for quad left, right and center. A little imagination will enable the systems designer to select characters which do not intrude too conspicuously into the text stream, and are nevertheless available on the printer's print chain or revolving drum. How much variety is available to him will depend, largely, upon whether his printer offers a character repertoire of only 64 symbols (including only upper case alpha letters) or whether it provides for 128. If the former, it will not be easy to ascertain whether the input codes in the text stream properly differentiate between capitals and lower case letters. Certain conventions, which impose a special burden upon the program and the hardware, have sometimes been used to reassure the reader that the input is in fact aware of this distinction. Printing a dot or period above or below the upper case characters is one method of achieving this. If the latter, he will not have to contend with the problem of differentiation between capitals and lower case letters, but he will still have to indicate

somehow, to the satisfaction of the proofreader, what words are in italics or bold, or other type faces, and he must also devise some sort of convention to describe characters that are beyond the capability of the print chain. For example, he may elect to over- or underprint to signal italics and bold, in somewhat the following fashion:

```
 .                          .         .
+DID YOU LISTEN TO THE SMOTHERS BROTHERS?


+Did you listen to the Smothers Brothers?


 *****************   !''''''''''''''
+THIS IS IN ITALICS.  THIS IS IN BOLD.


 *****************  ''''''''''''''
+This is in italics. This is in bold.


111111111111111111111111
THIS IS A NUMBER 1 HEAD


2222222222222222222222
This is a Number 2 Head
```

Special characters can be dealt with as follows:

```
                                        a  a
The French word for student is ealebve.
```

Or, if you wish to be assured that the correct accents for élève have been shown:

```
                                        a  b
The French word for student is ealebve.

                                       a b
The French word for student is eleve.
```

These features require extra programming effort, and they will slow down the throughput of the program, if the print run is provided at the same time that the hyphenated and justified output is produced. Upper and lower case printouts are generally slower than all capital printouts since the number of identical characters on the print chain is necessarily affected. Of course the printing of extra lines to generate symbols above and below the text line will mean more lines to be printed and will further slow the process. There is something to be said, therefore, for producing the print run off-line, or from a separate partition of core under multi-programming, the latter being the more practical solution if the magnetic tape or disk file output is to be formatted for further processing by the typesetting system.

### The Typeset Galley or Page

Proofreading is much more satisfactory if one works from the typeset image. Here there is no longer any question about the desired characters. They do not need to be represented symbolically. The convention of numbering lines for ease in making corrections can be satisfied by the output program and a feature has been provided on certain photounits to enable them to generate their own line numbers. The problem with this approach is that it takes more computer time and also places heavy demands upon the typesetting machine, which is no longer available, during such times, for the production of final output. However, at this stage in the development of the black art of computer typesetting, the chances of the initial typeset product being acceptable are not so great that one should discard the notion of setting the type twice (or more!).

If a typeset galley or page is to be provided for proofreading, the type need not necessarily be hyphenated and justified. It could be possible to skip this phase of the production process in the interest of reducing computer time. Ragged setting in lines of arbitrary length may prove more satisfactory than a printout, and the type size selected may be made with some regard for the proofreaders' eyes, even if the final page must be set in miniscule type to meet the customer's specifications.

Where such preliminary proof settings are part of the computer typesetting system, other techniques can be employed to assist the reader in his task. If there are special features of the job that require careful checking, these can be set in bold and in larger point sizes to bring them to the reader's attention. This may be particularly desirable where fielding or retrieval codes are embedded in the text and are not normally reproduced in the final typeset output, being present only for subsequent data manipulation purposes.

On the other hand, a case can be made for setting the job precisely as it will appear in final form (except for line numbers) so that a judgment can be formed of how it will look, how good the hyphenation is, and whether or not the overall size, and the individual components, meet the customer's expectations in a copy-fitting sense. If the job is to be shown to the customer at this stage such a procedure is highly desirable.

It may be seen that a case can be made for having several different options available, providing proof copy one way or another, depending upon the peculiarities of the job and the requirements of the customer. Pricing to the customer can reflect the costs of production under these alternate solutions.

### Proofreading on a "Scope"

Still another method of proofreading is available, and that is to display the interim output on the face of a cathode ray tube mounted in a terminal that is either on-line

to the computer or off-line. Some systems are designed with this in mind as the sole or major method of verifying and correcting. Others plan to use these "scopes" for the purpose of inserting the corrections, once they have been detected by conventional reading techniques.

### How Corrections are Made

Corrections made through the computer may be called "file updates" since they result in the creation of a new file, such as a new magnetic tape which contains the desired changes. The simplest way to visualize the process is to imagine that a magnetic tape file is mounted on a tape station and a paper tape containing corrections to various lines is mounted simultaneously on the paper tape reader. The program calls in the first correction message and ascertains that it applies to line 50. Consequently, the program proceeds to copy or "dupe" the first 49 lines from the original tape to an output tape. It then accepts the correction message, and having meanwhile ascertained that the next correction occurs on line 70, it resumes the duping process. Such corrections, to a magnetic tape file, must be made in serial order, and the correction program must be sophisticated enough to reject out-of-order messages, as well as to identify messages which cannot be accepted because of bad format. There must also be a printout or some other method of proving that the desired corrections have been applied, and to the proper position in the file!

Such correction passes, if they are properly written, take very little computer time. If the printer is suppressed, except for a printout of corrected lines, the corrections can be inserted virtually at a speed comparable to a tape dupe, somewhere in the neighborhood of 10,000 to 30,000 characters per second.

The major drawback to this method of making corrections is that there is no immediate feedback. By the time the reader receives a printout of his corrections he has forgotten the job and must familiarize himself again with its requirements, since the correction passes will presumably "queue" to be run along with other jobs. On the other hand, this is true of the reader's task under conventional typesetting techniques.

One special problem exists in making corrections to a computer typesetting file, and that is the imperativeness of getting not only the text revisions correct, but also the related coding, which may not be readily apparent, especially in a typeset galley. For this reason, the people whose task it is to insert the corrections (to write the correction messages) need to look at the code structure of the file to satisfy themselves that there are no hidden codes which may be lost during the correction cycle.

The use of an "editing" terminal—a correction device which is on-line to the computer and which can display the actual code structure both before and after the correction, is highly desirable, although this is another expensive item in a system which is probably already too expensive to be practical.

Such editing terminals usually access a disk file, and this makes it possible for corrections to be inserted in random sequence, rather than serially—a process which is highly desirable. Moreover, the entire file need not be "passed" in order to make only one or two changes which are widely separated in a large job.

### The Importance of "Trail" and "Back-Up"

When corrections are made by means of a computer system it is easy to lose track of the original input. If the corrections are misapplied the entire file can be destroyed,

and may have to be re-created from the beginning. The back-up tape or file must be preserved until one is certain that it is no longer useful. Also a record of changes is desirable to assist in diagnosis if something goes wrong (or to satisfy the customer's desire for evidence that his manuscript was really that dirty!) When changes are made on a disk it is perhaps easier to lose track of the history of the document, and special care must be taken that this record is not destroyed. It is less likely to be lost if the file was loaded on disk from a tape (which can be preserved) than if the file resides on disk until the job is completed.

### The Problem of Reprocessability

Probably the most perplexing aspect of the correction problem is that of the tape format itself. For some reason this writer has never been able entirely to understand, there are many correction systems in existence today which do not permit the reprocessing of the file (for correction or update purposes) after it has gone through hyphenation and justification. Something apparently happens to the format of the file in the latter process which makes it no longer acceptable to the correction routine. This means that corrections must be applied to the *input* file (input to the h & j program) rather than to the output file. Since the line numbers will no longer be the same it will be very difficult to find the location of the corrections, and much time is lost. This is one of the single most important reasons why computer typesetting of one-time projects (as distinguished from typesetting from data banks or in reiterative applications) is often tedious and uneconomical. A good system should be able to accept updates no matter what the stage of the job is, and it should incorporate those changes in random order with only the minimum of reprocessing. It is obvious that a revision to the text which adds or deletes words will affect line justification, the line count, and perhaps even the page makeup of the job, if that has already been accomplished. It is not necessary, however, to re-run the entire job to rectify the damage done in the correction cycle. It should be possible to re-do only that which has been undone!

### The Correction Language

Where corrections are made on a CRT terminal there is no "language" problem, as a rule, because the terminal "hardware" provides the computer with the information it requires. First the line or "page" will be called out. A cursor or a light pen will then locate the position in the text where the change is to take place. A button or key will indicate the nature of the change (whether a deletion or an insertion, for example), and then the associated keyboard will accept the text and code changes desired to be inserted.

But where correction messages are recorded by means of a paper tape keyboard, with the tape to be fed into the computer at a later point of time, or in an on-line mode from a typewriter terminal, some method of addressing the point at which the correction is to be made is necessary. (This is the correction "language.") Frequently this is done by addressing first a general text area, such as a chapter, block or "take," and then, within that area, a line number. Some relatively simple-minded programs then require that the *entire line* must be replaced, but a more useful solution is one which is able to locate the point within the line at which the correction is to be effective. This could be accomplished by numbering each character within the line, or each word within the line, or by looking for the occurrence of certain unique characters within that line. A correction message could read, for example:

15;03,05:word/

where the desired change was to insert "word" between the third and the fifth spaces on line 15, thus deleting whatever two words were there presently, and substituting one in their place. Some programs require the user to state whether he wishes to *delete, insert* or *replace;* in others the intention is obvious from the context.

Just as it is easy for the correction program to count word spaces or characters across the line, so it is also easy for it to search for the occurrence of certain unique character combinations. Thus a correction message could read:

15;mis:dis/

This would be an instruction to substitute the letters "dis" for "mis" wherever they occur in the line, so that *mistrust* could be changed to *distrust,* for example.

Where such character combinations are used, the reviser must be cautious not to introduce new errors inadvertently. Let the reader study the following possible consequence of a correction message:

*He said that this house was thisir own.*

Such a result could be produced by substituting "this" for "the" in the phrase: *He said that the house was their own.*

### Global Corrections

Another capability of some correction routines, which will be discussed in a later chapter on "editing" is the making of "global" corrections. This would be an instruction to change a certain combination of characters to another combination, "globally"—that is, throughout the job.

### Distinction Between Correction and Editing Routines

A correction routine can be used for the purpose of inserting author's alterations. Deletions and insertions, as well as substitutions, can be made quite simply, and whatever lines need to be re-justified as a consequence of these changes can either be processed as a part of the correction routine or, more likely, subsequent thereto. Obviously, a reprocessible file is most useful. In this sense there is no difference between "correcting" and "editing" except one of purpose. One corrects what one did wrong. One edits what one did right but now wishes to change. There are other aspects to editing, however, that will be considered later on.

### The Use of External Correction Techniques

Special configurations of hardware and software are sometimes assembled for the purpose of creating and revising input. These are particularly useful where they are available to the author or his secretary. A draft can be prepared, recorded on an MTST [1] cartridge, played back for retyping, revised on a subsequent cartridge, and finally delivered to the computer typesetter as input in lieu of manuscript. Or, the initial manuscript can be prepared on a typewriter with a typeface which can be scanned by an optical character recognition device, and revisions in that manuscript can be programmed into the scanning process if, for example, they are read before the page is read, remembered by the computer, and applied when the page itself is scanned and its contents written in binary format onto magnetic tape. A third alternative is to input on-line to a computer, and to call the text back from tape or disk, for subsequent corrections and editorial revisions. Time sharing terminals may be installed

[1] Magnetic Tape Selectric Typewriter.

in your own home or office for this purpose or you may purchase a mini-computer with several terminals for use wherever several authors or editors are at work.

### The Incorporation of Typographic Codes

Manuscripts which are supplied to the computer typesetter in machine-readable form by one of the above methods may or may not contain the necessary typographic encoding. If they do not, it will be necessary for the typesetter to use his correction techniques in order to add these highly important flags and instructions, or else to write a special program which recognizes certain peculiarities of the job as indicia for the location of typographic symbols. The latter is a special case not very readily adaptable to the processing of text material, but highly satisfactory if the data supplied in machine-readable form is "structured."

This application—the processing of structured data—will be considered further in Chapter XVII.

### Review Questions

1. Why is a computer correction routine more essential for a photocomposition system than for hot metal typesetting?
2. Why does the input have to be stored in order to be corrected?
3. How would you distinguish between "correcting" and "editing"? Is there any merit to making such a distinction?
4. In a technical sense, what is "verifying"?
5. What problems would a computer program encounter in trying to verify text from two input sources?
6. Why are diagnostic routines useful?
7. What are the arguments *for* proofreading computer printouts? What are the arguments *against*?
8. What are the alternatives to the computer printout as a vehicle for proofreading?
9. Why might several proofing approaches be desirable?
10. What procedure does the computer go through to "update" the "file"?
11. What advantages can you see in the use of a CRT display terminal for making corrections?
12. What do we mean by a "trail" in the correction procedure?
13. Why is it important to be able to reprocess the output of the h & j pass?
14. What methods are used in order to address the point at which a correction is to be inserted?
15. What is a "global" correction?

CHAPTER XIV

# Pagination by Computer

The dictionary definition of *pagination* has generally meant the process of assigning page numbers to a book, but by extension it also means the act of dividing a manuscript into pages. The term is used in the computer typesetting world to refer to the process of making up pages and outputting codes which, when they drive the typesetting unit, will cause the device to generate fully composed pages.

Computer pagination programs are not yet very well understood or very successful. Most of those who are engaged in computer typesetting produce galleys which they make up by hand, or produce type in pages only for very routine applications, such as technical manuals, parts catalogs or telephone directories. Most pagination programs are "special purpose" in character, being written to cope with the requirements of a particular job, large enough in terms of number of pages and frequency of appearance to warrant the application of computer programming time and talent.

Although one or two computer typesetting companies had developed somewhat more general-purpose page makeup programs in or about 1965 or 1966, the examples that come to mind for the most part grew out of the opportunity to take advantage of the capabilities of the Photon "ZIP." Here was a machine capable of setting wide-measure composition at high speeds—producing a volume of composition that could hardly be dealt with by hand stripping methods. Moreover, the work which was typeset by this hungry machine consisted mostly of unimaginative lists, derived from computer files. Most of it was of a tabular or multi-column character. In every case it involved the use of very few type faces, and all lines across the page were aligned, so that the ZIP could sweep across the page, laying down three or four columns of data, all to the same increment of leading. If occasional subheads were set in larger sizes these were capable of being expressed, with their associated white spaces, in the same common denominator. For example, if the job text were set in eight point solid, the subheads might be set twelve on eight, with two blank lines above and one below. Thus the optical white space above the head would be on the order of twelve points, with eight points below. This simple-minded approach was very congenial to the ZIP because all type sizes were predetermined by the stored photo matrices, since the machine has no capability of producing a range of sizes by means of its lens system.

Some of the early setting was not even "justified," and it resembled a computer printout, except that more information could be crowded onto the page due to the proportionality of the type.

There were two ways of setting multi-column material. One was to compose a line within one column and then "tab" out to the beginning of the second column. Actually, this was done most easily on the ATF B-8, because this typewriter-like device

had tab stops which could be set manually just as on a typewriter. Unfortunately, few machines with higher speed capability could claim this convenient facility, and it was necessary to devise a more elaborate solution. This required that every line be justified by the computer (and not by the photounit, unless it had a "multi-just" feature which was seldom if ever present). If all spacing elements were calculated by the computer program, and "put out" on the output tape, then the photounit could simply lay down fixed increments along the line of three columns, or whatever number of columns was specified, with additional units of fixed space (perhaps two pica ems) in the column gutters. Of course if one width value was short in the first column the other two columns across that line would appear ragged.

In substance, however, this method of composing multi-column material is still used today by computer programs that set across the page. A "column merge" program is all that is required. Text is first composed as in galleys. Lines are numbered or otherwise identified, and then the first line, the thirty-first and the sixty-second are arranged side by side by pulling them in an orderly fashion from core or from external storage. Perhaps the simplest way to accomplish this is to write out the first thirty lines (say) to tape station #1, the next thirty to tape station #2, the third thirty to tape station #3, and then the next thirty to tape station #1, etc. A program which calls a line from each tape, consecutively, and puts extra white space between the lines written from tapes one and two, and provides a film advance signal after writing the line from tape #3, will do the job very nicely.

But an additional degree of sophistication enters in when it comes to putting forth the page number (folio) and running head or feet. However, this information could be obtained from still a fourth tape, if the contents of such running heads can be known in advance. If they are "tell-tales," indicating, for example, the first name in the left column at the top of the left page, and the last name in the right-hand column at the right of the facing page, then the special-purpose program would need to know where to look to get this information, and what to do with it once it found it.

### Problems With Multi-Column Setting

One no sooner grasps the solution to the challenge of setting multi-column material across the page than he is beset by complications he did not foresee. When he pulls his lines from various tapes he loses the precedence code information which his typesetting unit requires. Consider, for example, the possibility that the first twenty-five lines of text are to be set in roman, and the next twenty or so in italic, and then the balance goes back to roman. Line 31, therefore, is supposed to be in italic, but the instruction to enter italic will not be encountered until far down the page. When it is encountered, then lines 26, 57 and 88 will *all* appear in italic! Obviously, the practice of giving the typesetting machine an instruction to go into a type face or size "and stay there until I tell you to come out!" will not work for this kind of setting—at least without some modification: the program that chops the text into columns must also "put out," at the beginning of each column line, what the photounit needs to know about that line. This is what the programmers at the University of Newcastle-upon-Tyne began to refer to as "lineinf."

### An Alternate Solution

If the photounit has "reverse leading" capability, the problem of multi-column setting is greatly simplified. Each column can be set independently, for the required depth, then the photographic film or paper can be reversed, and the second column

composed. The only "lineinf" required would be concerned with the displacement of the line across the page. A similar solution is possible (without negative leading) on a Linotron 1010 which composes the entire page on the face of the CRT, rather than to set one line at a time.

### Problems of Definition

Perhaps we were mistaken in plunging right into a rather complicated discussion of a particular problem of page makeup before providing some theoretical groundwork. However, the fact is that historically this is pretty much the way it happened—that is: one of the first pagination problems the industry tried to solve was multi-column setting, and the price it was prepared to pay at that time was to remain on uniform leading increments throughout the entire page. Of course it was believed that this made for better looking pages anyway.

It was not long, however, before programmers and their mentors began to yearn for the ability to provide more intricate solutions to the page makeup problem. Not all requirements were met by the simple capacity to lay down three or four columns across the page. In fact, one discovered that the business of composing a single-column page was surprisingly complex. Gradually it became possible to identify many of the problems, and to try to come up with acceptable solutions for them.

### The Not-So-Merry Widow

The most notorious problem is the *widow.* Programmers who had never before read a book, let alone appreciated its typographic niceties, suddenly learned, to their consternation, that to begin a page with a widow was not *comme il faut.* Nobody was entirely clear where this convention originated. Theirs not to reason why! Someone hazarded the explanation that if the first line on the page was short of the defined measure, as it would be in the case of most paragraph endings, then the page did not have the proper "frame." However, if the line at the top of the page consisted of a complete paragraph, which was only a partial line ("Here she comes now!" he cried.) somehow the frame argument was set aside.

It is not our privilege, the computer typesetters were told, to do away with these established conventions. Our challenge is to achieve the same solutions. Find out what it is that the hand makeup man does. Once you know what goes through his head when he divides galleys into pages, then you can write programs to do the same thing.

A *widow* can therefore be defined as a quadleft line that is immediately preceded by a justified line. If the preceding line was also a quadded line, then the line under consideration must be a stand-alone line which need not be rejected from a position at the top of the page.

Now that we can recognize the widow, we can avoid placing her at the top of the page. Unfortunately, we must then deal with the problem of having our pages come out with unequal depths. What is the rule about page depth? we asked. We learned that in the United States it was permissible for pages to be "a line long or a line short" so long as facing pages balanced, and so long as facing pages which were short were not followed directly by facing pages which were long (or vice versa). In Great Britain, oddly enough, it seemed that all pages, except chapter endings, had to come out precisely the same. When we asked an old-line typographer how this was achieved he said he couldn't rightly remember. It was jolly well done, but it had been so long since he had had to perform the operation himself, he had jolly well forgotten, don't

you know. He thought for a while longer and recalled that it must have been done by resetting some of the bloody lines, making them tighter or looser so that the "little lady" (the widow) got swallowed up. At that point the olive in his American martini also got swallowed up and he lost interest in the subject at hand.

It turns out, however, that precise page depths are often achievable if there are elements of vertical spacing in the page. An odd amount of space separates text from extract and extract from text. This can be stretched or shrunk a bit. There is room for a little "fiddle" (an Anglicism again!) above and below heads or around illustrations. Usually this does the trick. But woe unto you if these elements are not present to enable you to perform "vertical justification"! That term became very popular, for one suddenly saw that the problem of coming out to a fixed depth vertically was precisely the same kind of problem as coming out to a given width horizontally. There are fixed spaces (the lines themselves) and justifying spaces (the extra white space). And just as compositors sometimes "cheated" by indulging in letterspacing (space between letters as well as between words) so they were tempted to "cheat" by "carding"—adding small increments of space between lines of text, as one might do by putting cardboard strips between metal type slugs so that forty-one lines took up as much space as forty-two, and the final line was pushed over to the top of the next page in order to drive the widow down.

Could the photounit "card" discretely? Some could and some could not. Increments of an additional half point between some lines (starting from the top down or the bottom up) provided too crass a differentiation, even if the typesetting machine could advance its photographic film so minutely. Later, machines came along that could do this in tenths of a point, or even in thirty-seconds of a point!

"Carding" was not a sufficient (if acceptable) answer, however. Some pages simply couldn't be carded out, and even if there *was* additional white space for vertical justification, too much space would be required to bring the page to the proper depth. Why did this problem suddenly come to light? It was because there were certain instances, in the text stream, where it was not permissible to break the page even in the absence of a widow! What are some of these conditions?

When a head or subhead occurs it is customary (indeed, it would look odd otherwise!) for two or more lines of text to follow before the page can be broken. If a table appears, the table must be kept as one unit and not split over two pages. If the author waxes poetic and makes his point by a stanza of verse:

Our author would surely explode in a rage!
If *this* line broke to the following page!

One assignment Rocappi encountered fairly early in the game was a listing of lawyers, with their titles, birthplaces, and legal pedigrees:

Jonah W. Jones
Senior Partner
Hickory Rock, Pa., 1878
Admitted to practice before the Supreme Court of Pennsylvania in 1905.

It was not permissible to disassociate Mr. Jones from his title, nor from his birthplace. You could put his legal pedigree in the following column on the same page, or on the following facing page, but not on the obverse side of the same sheet.

It became evident, therefore, that there were "no-break" areas of composition. Two problems arose. One was to recognize these areas (how to tell a *head* from text, or a rhymed couplet from parlor conversation) and the other was to make pages come out even despite the lumpiness of the elements one had to contend with.

Tables and illustrations had to be recognized as coming within one of two classifications. Either the tables (or the white space for illustrations) had to be positioned precisely where the text required them to be (in which case these became "no-break" areas) or else they could be "floated" to another position if they would not fit directly beneath the text that referenced them.

Footnote call-out references and the accompanying footnotes also posed very serious problems. If the reference occurs on the page, generally speaking the footnote also must be placed at the bottom of that page. There are times when the footnote is so long that without it (and the line which references it) the page will be impossibly short. With both elements, the page becomes too long, and the footnote, which could even be larger than the entire allotted page size, must be carried over to the bottom of another page.

Our discussion of these difficulties, expressed as it is in the past tense, is not intended to suggest that these matters have now all been resolved. Far from it. But the major decisions have now been identified, and some program solutions have been developed. Perhaps the most complete listing of pagination elements is contained in a document put together by a task force under the chairmanship of William Tunnicliffe of the organization which is now known as the Graphic Communications Computer Association of PIA (the publisher of this book). Members of the association may obtain the report and, if they are engaged in a programming effort, they would be able to use it profitably.

### Conceptual Solutions

There seem to be two approaches to the challenge of devising a general-purpose pagination program (as differentiated from the special-purpose programs that may well be most effective for large and repetitive jobs, especially those containing structured data). One approach is to take the page depth as given and invariable, and to pour text into that page, regarding it as an *area,* and attempting to fill it up while hyphenating and justifying. A *trial setting* mode can be specified and if a widow should appear at the top of the next page, or a copy block emerges which cannot be broken, then the program will make an effort to resolve the problem by varying the areas of vertical justification, including carding, if that is permissible. If no solution can be found by these means, then a new definition of allowable interword space will be brought into play, in the hope that tighter or looser lines will provide an acceptable resolution of the problem.

In making up pages by this technique there are still other solutions possible, since the program is capable of receiving instructions in a language which has been devised especially for this purpose. We are, of course, discussing the concepts that Dr. Michael Barnett introduced into the RCA PAGE-1 program. Quoting from an advertising brochure published by RCA, entitled "PAGE-1, The language of electronic type composition," we read:

> One of the most valuable capabilities of digital computers is their ability to choose alternative courses of action. This capability has been harnessed in PAGE-1 through the inclusion of a set of "conditional statements." Each of

> these statements tests two specified parameters to see whether there is a certain relationship between them (e.g., equal, not equal, greater than). Depending upon the result, one of several alternative actions can be taken.
>
> The PAGE-1 language also provides complete arithmetic facilities: addition, subtraction, multiplication, and division.
>
> These conditional and arithmetic statements enable the computer to make decisions similar to the ones human compositors must make . . .

A program strikingly similar to that of RCA, is known as "Ultra-X" and it was developed by a company known as PICA. (See *Dictionary of the Ultra-X Composition Language, Universal Language for Typographic Reproduction Applications* by PICA, Printing Industry Computer Associates, of Princeton, N.J.) The reader will wish to study the command language in this booklet, including such specifics as *left boundary, page depth,* and especially *whenever bottom* and *whenever right.*

A publication of *Composition Information Services* (CIS Installations & Applications, November 1970) makes the following comment:

> The PICA programs which CompuComp uses, like the RCA Page 1 and Page 2 programs, and the Harris HCL system, are what has been called "typographic compilers." They provide a comprehensive "language" that allows the user to assemble a "program" of system commands that will compose and make up text material according to specification. Each new application thus requires a new program to tell the typesetting system how to process the material at hand.
>
> According to CompuComp's vice president of marketing, Irv Roberts, "At the present moment, the economics of developing these 'one-shot' programs rules out computer typesetting for a majority of current typesetting applications." . . . . . .
>
> This brings out into the open a controversy which has been quietly smoldering ever since RCA started work on its "Page 1:" Notably, can a typographic compiler be used efficiently for this kind of work, and if so, how? CompuComp evidently feels that it cannot . . .
>
> There are, of course, many other firms—principally those who have written their own software—who are accomplishing full page make-up, and who are doing CRT typesetting without using typographic compilers, but since the major programs which are commercially available for this purpose are compilers, the questions that CompuComp has raised are very meaningful.

Returning to the RCA brochure, however, we read:

> Actually, all type composition can be reduced to blocks. By calling in several blocks by format statements, you can compose an entire page.... And you can easily write as many of these prestored formats as you need to define all your standard composition requirements.

We do not wish to become embroiled in the controversy as to whether or not compiler programs for page makeup are effective. We have seen some very complex and attractive books produced with the use of this language. We do not know at what cost, in terms of human effort, ingenuity and dollars. It is true that the compiler language is very "powerful." But it must be used, especially on new applications, by persons who are thoroughly familiar with it, as a tool of their trade.

An alternative conceptual solution is one which attempts to solve the problem of page makeup on a chapter-by-chapter basis. It provides all of the elements which the computer program requires. Some of these are of course identified on the input. Others are specified by pagination parameters which are unique for the job. These parameters are not conditional statements, however. They indicate the desired page depth, the number of columns, the number of lines that must follow a head or subhead, the treatment of chapter openings, and the extent to which white space can be stretched or shrunk to help the page fitting process. Carding is either allowed, subject to certain limits, or it is prohibited. The amount by which facing pages can differ from succeeding or preceding pairs of facing pages is also specified.

The problem of making up pages to follow these instructions is then assigned to the computer. All of the conditional elements have already been written into the program. The program attempts to deal with every possible circumstance and tries to get the best answer. It should succeed in reaching a solution as good as, or better, than that which would be reached by a skilled makeup man who is confronted with precisely the same restrictions. If it cannot solve the problem it can call for "help" and it may be that the pagination requirements will have to be relaxed to deal with a specific difficulty, even if it is not felt appropriate to relax them as a general proposition.

It would even be possible to give the computer program the right to put into effect an exceptional degree of relaxation, however, under the same conditions that the human would exercise this judgment.

As we have indicated, pagination programs of a general nature are relatively new and still largely experimental. They are by no means fool-proof. Certain kinds of jobs are best tackled by hand makeup or by the development of detailed computer instructions to nurse the job through. The problem is not whether programmatic solutions can be found, but rather how efficient the programs are and what size computer is required to run them. If a chapter-by-chapter pagination solution is desired, as opposed to a page-by-page approach, and if this means that proposed pages must be held in memory or in peripheral disk storage until the chapter is released, then there are heavy demands upon the hardware, in terms of its storage capacity. Solutions which minimize the storage problem are proprietary and details cannot be supplied in this study.

**Pagination Solutions for Newspapers and Periodicals**

Very little has as yet been done in this field. For the most part galleys are stripped into pages. However, we have seen the results of a page makeup program applied to periodicals by the R. R. Donnelley Electronic Graphics Division, and we have been very favorably impressed. On the whole, it is our impression that the task is a relatively simple one, since we believe that editors will still wish to lay out their pages in galley format, and the page makeup process becomes a simple matter of writing up instructions to the computer on what sequence of lines to pull from the file and where to position them.

Ultimately, perhaps, page makeup can be achieved by specific human instruction, while the editor studies the proposed page on the face of a display tube. He can then specify a different layout and cause the text to be spilled into the areas he indicates, rehyphenating and rejustifying as necessary. The problem at the moment is that such display devices do not have the capability of representing the entire page (even of a straight forward novel) in the type faces desired, or even in proportional type. Consequently, the page can only be presented to the editor symbolically or in the form of diagrams or outlines. The Federal Systems Division of IBM, at Gaithersburg, Maryland, has devised such a technique for the display and re-arrangement of text for a Japanese newspaper application, the identity of which, while generally known, has not, so far as we know, been made public by IBM.

At the moment we are not aware of any programs for English language setting which provide an interactive and re-entrant capability for page makeup, or even for hyphenation and justification. A terminal-oriented program, which permits the user to talk to the computer in "real time" so that the computer responds instantly to his commands, requires different programming concepts from those of the "batch processing" variety.

It is in the area of pagination, especially of the interactive variety, that we see the most challenging problems ahead of us. The only comparable area, also exciting to contemplate, lies in the extension of the use of the computer as a text editing tool.

**Review Questions**

1. Would it be possible for a photounit to justify a line which is made up of three separate columns, each itself being justified? If not, how else can this be done?
2. What problems are encountered in merging input in order to set multi-column material across the page?
3. Can you think of any other way to set multi-column material than to set it across the page?
4. What is a "widow"? How can you avoid widows and still maintain the desired page depth?
5. What are the elements in a page that permit some degree of "vertical justification?"
6. What advantages are derived from the ability to move lines back and forth over an entire chapter as opposed to solving the page makeup problem within the confines of a specified page depth and without being able to "back up" and re-do the preceding pages?
7. What advantages are derived from the ability to "trial set" the page and then to hyphenate and justify all over again if necessary?
8. Do you understand the difference, in principle, between a "compiler" program and one in which all of the instructions are available to the computer? How could conditional statements be used in a page makeup program? What alternative way would there be to solve the same kind of problem?
9. What are some of the problems in developing an approach to pagination which enable the editor to make the fundamental paging decisions on a CRT display console?

CHAPTER XV

# Input Theory

## Modularity

One major reason for departing from a one-pass system of computer typesetting is in order to create modular programs. If the input program is a separate pass (or at least a separate module) it is easier to "plug it in" to the hyphenation and justification program. It is also easier to revise it, and modularity permits the use of more than one input program to accommodate more than one input device.

Additional input programs become necessary as new input keyboards are developed, or as alternatives to keyboarding come into being. It is too much to expect that all new approaches will offer code-for-code compatibility with older solutions. Moreover, one advantage of such newer approaches is to simplify the input task by reducing keystrokes, so that code-for-code compatibility is not really desired.

## Code Correspondence Not Essential

There need be no direct correspondence between the input coding and the internal coding of the computer itself. And certainly there seldom will be an identity between either input or internal coding and that required of the output device. Furthermore, modular design permits the existence of several output programs for several different typesetting machines.

We must understand, however, that differences between input and internal language, for example, can be more substantial than mere differences in coding structure. The *structure* of input may differ from that of the machine language. What is a single code in the input stream can be translated into a string of codes within the computer, just as this may also be true as between computer and output device. Conversely, a *string* of input codes can become *one* single symbol within the framework of the computer's internal character set.

This capability of translating not only one code to another, but one string of codes to something else, offers great flexibility to the solution of input problems, and permits a variety of different approaches to be employed, even by the same user, in order to meet different needs.

For every application there is, presumably, an ideal solution, but it will have to take into account factors of human convenience and habit as well as ideal theoretical conceptions. A keyboard with 256 keys may or may not be a good idea. If the layout bears no conceivable resemblance to any other keyboard, and requires substantial retraining it probably would have to provide a demonstrably better solution to be worth the effort. On the other hand, it may in fact be easier to use a keyboard with

ten keys, all of which rest immediately under the fingers (something on the order of a Stenotype), and to strike combinations of these ten keys, rather than to search for the proper key on a larger keyboard.

A good solution will be oriented toward the user's specific applications, and yet there should probably be some degree of commonality between all approaches: they need to be reduceable conceptually to some basic structure which will facilitate data interchange. In order to achieve this objective we will need to reach some kind of agreement on computer conventions for typesetting and related information interchange. Conventions, we are reminded, need not be binding. Yet there are times when it is essential to be certain that we are understood. David Frost has suggested that a proper young lady in the United States (if there still are some!) would hardly leave word with the hotel room clerk that she would like to be knocked up at eight o'clock in the morning! Such a request in Britain might be disappointingly commonplace, and not because of a difference of *moral* standards. It is the use of the conversational (information interchange) idiom that causes the confusion. Our own mental computers happily make allowances for some of these contemporary idiosyncracies, invoking a translate routine which can cope with phrases and words as well as individual graphic symbols. If we wish to write for posterity, however, we had better know which of our current groovy expressions future generations will be able to dig and which ones will bomb out.

Our need for conventions relates to two primary areas of information interchange. One has to do with the presentation of the character repertoire itself. The other is concerned with a definition and identification of typographic "situations."

Let us look first at the problem of the character repertoire. If we can agree to group certain characters into families or "sets" we can greatly simplify the problem of character access. For example, there are usually 44 keys on a typewriter, plus a space bar, and certain "function" keys: backspace, shift, tab set, tab clear, tab and margin release or reset. If our typewriter is a Selectric, with a "golf ball" element, we can design our own golf ball, and access any characters we want on that new golf ball if we know what key it is we need to strike. We might have a Greek element, and another for mathematics, and still another for foreign accents, and perhaps one with all kinds of interesting special characters we might need for scientific or logical representation. Suppose we worked out ten different golf ball elements. Apart from our desires with respect to type face design and type size, we could probably cover most of our typographic needs merely by mounting such new typing elements. We would soon learn which character set was which, and which keys to strike to access the desired symbols. But if everyone designed his own character sets, with entirely different groupings of characters, each positioned differently, moreover, on the typing element, it would be tedious indeed to locate the characters we wanted to use. We would neither know which golf ball to mount nor which key to strike.

It is true that a computer can cope with this problem better than a mere human, but for every different user it would be necessary to input an elaborate translate table, especially made up for that one application, listing every one of perhaps 880 different characters, and identifying the particular golf ball and the particular position on that element for each such character. Since the character repertoire is open-ended, the translate table routine would also have to have an infinite capacity for expansion. How much better it would be to agree on at least a half dozen families or groupings of characters that would mean the same for everybody, even if they did not accommodate each individual's needs quite as readily as his own special-purpose solution!

If such groupings or sets could be developed, what size should they be? We conceive of certain codes or characters as cutting across all character sets: these are flags or functions, and spacing elements which are needed at all times. But we think that 88 characters in an interchangeable set would be most useful, merely because the typewriter is so ubiquitous. We might say, "No, the typewriter is inefficient. Let's design our character sets on the basis of a keyboard layout of 100 or 128 different keys." But would it not be better to try to establish a convention on the basis of the *smallest common denominator?* We suspect the typewriter keyboard affords this opportunity, and its relevance is even more evident when we consider it as a machine for the creation of OCR input.

### Definitions of Terms

As a beginning to the establishment of conventions, a greater degree of precision in our vocabulary would be helpful, when it comes to describing the *things* that we must manipulate. In an extremely important paper prepared by Mr. Walter Jessel, of IBM's Systems Development Division in Boulder, Colorado, entitled "Extended Character Sets in Systems With Photoprinter Output," draft dated 1/21/70, we find the following helpful definitions:

> *Alphabet.* A set of letters or other characters with which one or more languages are written.
>
> *Sign.* A letter of an alphabet, a figure, or a symbol standing for one orthographic identity, independent of graphic appearance or size.
>
> *Text.* A purposeful concatenation of signs.
>
> *Graphic.* A representation (form, shape) of one sign (including space) contained in a font.
>
> *Character.* The basic unit of a font; a general term for both "sign" and "graphic," making no distinction between content and form.
>
> *Orthographic Equivalents.* Graphics of different design or size which would not alter the meaning conveyed by text (e.g. a word) in which they are contained if they were substituted for each other. *Antonym:* Orthographic Contrast.
>
> *Font.* A complete, purposeful assortment of characters for the composition of text in one alphabet, containing one and only one graphic for each sign. (The term *font* is deliberately restrictive to obtain unambiguous access to each member of the set from a single code, or through prefix conventions from two codes. One font may include upper case, lower case, and small cap letters of the alphabet, in addition to figures and symbols.)
>
> *Font Section.* A program-addressable subset of one font, containing 191 or fewer encoded graphics, with optimally ordered code assignments conforming to standard provisions. Not all fonts will contain all sections. Some sections may be common to several fonts.

> *Graphic Escape Code.* A control character, which, when preceding a sign code, causes that code to represent a different sign. A mnemonic relationship will usually exist between the original sign and that produced through Graphic Escape. (E.g.; Spanish open exclamation point = Graphic Escape + Exclamation Point.) A graphic escape code may be generated on an input device by a special function key, or by a key whose original function was given up.

This writer is not entirely in agreement with some of the suggestions Mr. Jessel advances in his paper, but he finds the terminology extremely helpful. For example, the letter *a* is a sign, whether it is roman or bold or italic, in eight points or eighteen. The letter *A* is another such sign. On the other hand, a Times Roman eight-point *a* is also a *graphic.* An italic *a* and a roman *a* are orthographic equivalents. Our major point of difference lies in the fact that we believe that a *Font Section* should consist only of 80 characters, and the Graphic Escape Code cannot, in our judgment, meaningfully and usefully be followed only by a single character. The mnemonic or symbology must allow for greater variety, since the characters must presumably be selected from the basic font section.

Actually, however, Mr. Jessel doesn't quite come up with the definition we feel to be vitally essential. *Font Section* relates to a purposeful selection of particular graphics. What we need is a term which relates to a purposeful selection of a standard subset of *signs.*

### Differentiating Between Signs, Graphics and Non-Signs

The other problem with Mr. Jessel's analysis is that he does not give us the badly needed term for the characters (*not* signs) which serve as flags or functions, or field deliminators, or commands. He gives us only one such term: Graphic Escape Code. It must be obvious that there is a need for a number of such characters which are non-signs. Presently there are codes designated for use as file separator, group separator, record separator, unit separator, end of block, field separator, tape mark, end of transmission, and other similar characters which have been appropriated somewhat wastefully.

Our first task should be to differentiate between what we shall call *signs,* and *non-signs,* or data codes. Then, within the *signs* there will be found to be a small group which would be either universal to all subsets or would be independent of any subset. Just as the non-signs would be universal, so would these particular signs, and these are spacing codes—the em, the en, the thin space, a variable fixed space, and a no-break justifying space as well as a regular justifying space. If Mr. Jessel prefers, we could call these non-signs as well, since spaces may not be properly termed as *graphics.* The Graphic Escape code, and the other similar typesetting codes, such as quad left, quad right, and quad center, could also be considered as one sub-order of data codes. But at least one of them must be appropriated from the typewriter keyboard (and for this purpose we suggest the asterisk).

All other signs will consist of symbols which have an orthographic identity and which will of course have orthographic equivalents. Of these, there will be *one basic*

*subset* which will correspond with a typewriter character repertoire. All other subsets should be standard and interchangeable, except that the number of such standard subsets should be open-ended, and it should be possible for the same sign to be found on several subsets, to avoid inconvenient shifting.

It will be necessary to distinguish, in the input stream, between one subset and another, where there is no orthographic equivalency, and between one graphic and another when they are orthographic equivalents. One kind of flag should indicate that we are merely switching graphics, but not signs. Another kind of flag should specify that we are in fact switching signs. We may be in the same font, but in a different sign subset (to get Times Roman small caps) or in the same sign subset but in a different font (to get larger point sizes and display letters).

Escape symbols are therefore required. The symbol will identify the fact that an escape is desired to a different *sign* or to a different *graphic,* or perhaps to both. This can be done by using the escape sign (which will be our asterisk) followed by one order of signs to indicate a shift of graphic; another order of graphics (which must be found within the basic sign subset) would indicate a shift of sign subset. Wherever these particular characters or codes appear (when preceded by an asterisk) they must be understood, in effect, to be universal and independent of any subset of signs. We can therefore use *1a to designate the basic sign subset in the particular graphic font which we may from time to time choose to designate as *a.* The signal *2 will call for a shift in sign subset, from "layout" #1 to layout #2, but since it is not otherwise stipulated, we would remain in font *a.*[1] If we are in subset #1 and the signal carries us to *b, which is a different font, we remain in subset #1. On the other hand, *2b subset carries us both to a different subset and a different font.

The sign group #1 will always mean a "character set" which is readily identified and which brooks no substitutions. I will know, for example, that input code 0100 1001 is now to be considered to mean the lower case letter *i,* and not a sign for a rhomboid parallelogram, or a right angle, a radical root or an Indian rupee. If my program keeps proper track of my sign language it will know whether to submit the "word" to hyphenation logic or to give up on it in despair.

We should make it clear, however, that the standards we are seeking, or, more properly, the *conventions* that need to be adopted, will not preclude other solutions on the input end of the system. Any character that is needed frequently for a given job can be assigned a keyboard location for that job, or can be given some kind of tag for that particular job, so long as the input translate routine can be instructed to convert it into something understandable to the system. Again, *there needs to be no standardization on the output end.* Typesetting machines will differ and groupings of signs and graphics may be arranged within the typesetter for reasons of job efficiency. It is the integrity of the internal classification scheme—the taxonomy of information representation, if you will—that needs to be safeguarded. These are the distinctions that the program operates upon. And it is these that will need to be passed on to other users. Translated they may be, for the convenience of another system, but their taxonomy will not be jeopardized in so doing.

### Implications to the Input Device

Our system will not be device-limited, but it will be oriented toward the typewriter as the common denominator of all information-transfer machines. This will make it

[1] We propose to follow with a *z* wherever font remains unchanged. See Chapter XVIII.

possible for the output of the typewriter to be scanned and converted into machine-readable input. While keyboarding on the typewriter will not be as efficient as it would be on a keyboard with special-purpose keys, it will nevertheless provide a vehicle which will be able to access the infinite character repertoire that typography has to offer.

### Scanning the Typewriter Font

For typesetting purposes, optical character recognition is often applied in three different ways. One is to attempt to capture documents which have already been set in type. A second is to attempt to capture documents which have already been created on the typewriter. A third is to capture documents which will be created on the typewriter for the purpose of being captured. It is this third category we have in mind at this point. We have elected to discuss it because it also represents most clearly the problem we encounter in conveying information to the computer when we are confronted with a limited keyboard and, conceivably, also, a small gamut of codes to work with (such as a six level paper tape).

Immediately we will have to make up our mind that combinations of keystrokes will be required to represent single non-signs and the non-graphic signs for the different kinds of spaces. A bigger keyboard can eliminate the need for such combinations. But the problem can nevertheless be resolved on the typewriter keyboard.

We shall endeavor to illustrate this in order to demonstrate the manner in which input can be converted to the system language which the program presumably has been written to implement.

We have already decided to surrender the use of the asterisk as the single flag or signal code we shall require. Our input solutions could be much simpler, and involve much less input keyboarding if we appropriated *two* positions for *two* flags. We shall show how both solutions might be implemented.

We are going to need space codes which cannot be created by our typewriter. A non-graphic will be assumed to be a justifying space. We may use **e or ¢e for the em space. Similarly **n or ¢n may represent a nut space, and **t or ¢t can provide the thin space. A variable fixed space will be helpful. This is a fixed space whose width value can be defined from time to time. The codes **v or ¢v can describe this condition. Notice that all of these rather awkward strings could be indicated by one keystroke on a larger input device. They will be translated as only one code in the internal logic of the hexadecimal computer coding. Similarly, **l, **r and **c can be used for quads left, right and center, or, alternatively, ¢l, etc.

When we wish to move from the basic sign subset of the typewriter, we can call forth the change by *2 or *3, as described previously. A change in both sign subset and graphic is indicated by *3s, for example. We have had to use the double asterisk, or the cent sign, in order to avoid confusion between the space and quad codes, on the one hand, and the sign and graphic codes on the others. We should remember that there are other typographic commands. Some of these are quite complex, such as for levels of indentions. Two problems must be resolved: how to input these commands, and how to represent them in the internal language of the program. If the indention command is to be: **i or ¢i and we wish to suggest different levels of indent, we really need something like this: **ia for "begin indent level a." But all of the other double asterisk or ¢ codes are followed by only one keystroke's worth of instruction. There are

two solutions to this problem: one is a sub-routine (possibly on the input pass, possibly within the basic typesetting program itself) which, upon discovering that the operative character is *i* looks to see what level of indent is needed. The other solution is to use delimitors. Some typesetting programs feel much more comfortable using delimitors around all commands so that they do not get confused with text. This means appropriating the parentheses (or brackets), and it would mean giving up two additional characters on the typewriter subset and representing them only on an alternative sign subset. But with delimitors one can avoid confusion in proofreading hard copy. Hence (**ia) or (¢ia) would serve. Here is a printout simulation showing three different ways of representing the same set of instructions:

```
**ia**eThe computer printout.**l

¢ia¢eThe computer printout.¢l

(¢ia¢e)The computer printout.(¢l)

      The computer printout.
¢cia¢e                      ¢l
```

It is unfortunate that the letter *l* (ell) and the numeral one look so much alike. The instructions say: give me an indent level *a,* follow it by an em space. At the end put a quad left.

### Keyboard Longhand

We shall discuss keyboard shorthand techniques in the following chapter. But keyboard longhand can be an input solution when one does not wish to find the sign subset for a particular sign that is needed, and one knows that, although the input will require additional keystrokes, this will be simpler than stopping to locate the sign and its subset and position requirements. A three-letter mnemonic might describe it, and the input program might then find it. We might keyboard ¢¢inf for *infinity,* and ¢¢deg for *degree,* ¢¢upa for *up arrow,* ¢¢dna for down arrow, etc.

We can see, therefore, that the input pass can serve not only to translate the codes from the input machine into the language needed by the typesetting program, but it can be devised to maximize efficiency on whatever input keyboard or OCR method may be chosen by translating a long string of easily-conceived codes to a few simple internal codes, or a short sequence of input codes to a longer string of computer instructions. Such input programs are sometimes known as "pre-processors" since they translate and interpret, as well as reformat, so that the basic program itself can cope with the input.

One other notion may be injected to complete the terminological considerations that are relevant to an understanding of this highly complex field. This is the concept of a "locking" and a "non-locking" escape sequence. Normally, a shift to another sign subset or to another graphics font is to be presumed to continue until further notice. You are, in effect, "locked in." But certain non-locking or self-cancelling conditions can be recognized by the program. It has been suggested that an *accent* is such a condition. When the shift is made to a subset containing only accents, it may reasonably be assumed that after the next character has been designated, the previous sign

subset will govern. There may be other circumstances in which the non-locking escape convention can be utilized.

### Is Such an Input Structure Intelligible?

The wider the range of keys and input codes that can be used the easier it is to provide reliable input, so long as that range of keys can be made familiar to the operator. Proofing problems, which we have already considered, of course rise to haunt us: is there a hard copy representation for the key which is struck, and if so, does it consist of a single unintelligible code symbol or a string of unintelligible alphabetic characters? Whichever it is, it will make sense only to the initiated.

What alternatives are there? The truth is that there are really none, at least at the moment, if one wants to produce work which is highly complex, which requires frequent shifts into different sign subsets and different graphic fonts, and if one wishes to invoke a large sequence of typographic commands. On the other hand, there is a class of work which demands little in terms of character repertoire or format which could no doubt be handled quite simply.

If this discussion has seemed to be too abstract, let the reader remember that one of the major challenges which shall face us in the near future is the task of establishing conventions for information interchange. Unless we can contrive to do this we shall not use our new technology advantageously. We should indeed be obliged to abandon our undertaking much as did the workmen at the tower of Babel.

### Review Questions

1. See if you can explain how an input pass, or a pre-processor pass can convert one character or code into another, and one string of commands into another, or one code into a string of commands. Then explain how a string of commands can be converted into one code.
2. Do you understand the distinction between a *sign* and a *graphic?*
3. Draw a diagram in which you deal with alternate subsets of signs which are standard, as opposed to alternate subsets of signs which are not in standard locations. Imagine that you are translating between a set of standard subsets and a set of non-standard subsets.
4. Why does the writer feel that the typewriter keyboard should provide the basis for making a determination as to the optimum size of the sign subset?
5. How are the necessary typographic command and spacing codes to be generated?

CHAPTER XVI

# Keyboard Shorthand and Macro Instructions, or Global Notations

## Definition

*Keyboard shorthand* consists of the ability to simplify tedious keyboarding by substituting a unique string of a relatively few input characters and subsequently interpreting or translating that short string into the longer element that it stands for.

One definition of a *macro instruction* is a "source language instruction that has the capability of generating more than one machine-language instruction." [1]

In this chapter we shall describe a technique that accomplishes the former, and is also something like the latter, but is more than either, because it is *redefinable.* In the industry, the term "macro" is sometimes used to define what it is that we have in mind, but this term is technically misleading, since we are not truly dealing with a source or programming language. Yet something more than shorthand is involved, and the redefinability of the character string is one of its happiest features.

## Pure Shorthand

Let us begin with the simple concept of shorthand. We notice that there is a relatively frequent occurrence of a string of characters that is time-consuming and perhaps difficult to input. We therefore devise a unique and purposeful combination of only a few letters; we inform the program that this smaller string shall be taken to mean the larger one. Often the letter *q* is used as the first character in the string, and may be followed by any second (or second and third) character so long as the second is not *u, a* or *b.* (The *u* is obvious; *a* sometimes follows as in the word Qantas, and *qb* will be found on the sports page.) The use of *qca, qcb,* etc. will provide such a unique string, and many variants are possible. One way to use this method of shorthand input is to define the true character string the first time the shorthand is used. This could be done, for example, by using *qq* as a warning that a term and its definition are about to follow:

*qq*Character string*qca*

informs the program that the words *Character string* will hereafter be designated as *qca.* In some applications the definition is not in fact provided at that time; the person who decides to use the shorthand string simply uses it and makes a note of what it stands

[1] Charles J. Sippl, "Computer Dictionary and Handbook," Howard W. Sams & Co., Inc., May, 1966, p. 181.

for. The program will be instructed subsequently. In other cases the substitution is not *global* (that is, it will not last throughout the job, but only until further notice). Perhaps a different flag is required for the non-global as opposed to the global substitution. If we should be inputting a telephone directory and we encounter:

Miller, Arthur H.
Miller, Charles F.
Miller, Joseph R.

and so forth, we may decide to use a non-global form:

@@Miller, @@Arthur H.
@@Charles F.
@@Joseph R.

where the double "at" signs surround not only *Miller,* but also the comma and the space which follow. The definition is non-global because later on we will do the same thing, using the same double "at" signs with Murphy, Nelson, and other names whose occurrence will be listed frequently.

Before computers were in common use, we saw a "black box" employed to generate repetitive names, in much the same manner, at the Union Printers' School of the International Typographical Union. It had been developed by one of their research staff in the mid-1950s, specifically for telephone work. It now appears as a feature of a good many "edit" programs provided by time-sharing services, but it can, of course, be incorporated into any input program.

### The Format Statement

A somewhat similar concept consists of the stored *format statement.* This is a feature of the RCA PAGE-1 program and it was devised to simplify and to generalize the rather elaborate input structure that program requires. Normally one would precede a type face with a definition of that face (tf,l;ps,8;bl,10) which means *type face #1, in eight point with base leading of 10.* Subsequently the measure would be specified, and perhaps some other features, such as paragraph indentation, ragged setting and the like. All of these could be defined in a format statement (fs,A1) gives a tag of *A1* to a particular set of codes used to describe that particular condition. Whenever that set of conditions is desired, the format statement can be brought in by the use of its label. Moreover, *it will be possible to redefine that format statement, and assign other type faces, point sizes, body leading, width measures, hyphenation conditions and so forth.*

### Global Notations

We have chosen to invent the term "global notations" to describe the use of unique strings of codes, which are capable of definition and redefinition, which will apply globally to the job in question, and which extend the use of format statements so that they can stand for text, or command codes, or combinations of both. Initially, we were disposed to use the term "Macro Instruction" to describe this concept, and in some of the prior articles we have written, or presentations we have made, this phrase has been employed. However, the expression is misleading, since the macro concept is a program concept, and what we have in mind works equally well with text strings and, as we have said, combinations of text and commands or tags. Global notation is the best name we can think of to identify this feature, which is one of the most useful tools available to the computer typesetter or his customer.

### When the Substitution Occurs

There are occasions, when one uses keyboard shorthand or global notation, that the substitution into the intended string can and should be done in a preprocessor or input pass, but there are advantages in delaying this substitution until later in the game. One advantage is that the use of small strings is not only easier to keyboard, but *also to proofread.* Consequently, the readers or verifiers should be taught to work with these notations, with the understanding that the definition which will later be substituted will, of course, be carefully verified.

Another point to be made is that, even when the substitution is made, *it should be reversible.* In other words, the global notation should not disappear from the file, but it should be disregarded until it is able to play another role in the process. In the meantime, the substituted string of "real" characters can be employed, but they should be made to disappear (or some of them) in favor of the initial notation, when they have served their purpose. This keeps the file in its most general and most abstract form, and permits its use in more flexible ways than otherwise would be the case. In fact, a case can be made for never permitting the redefinitions to appear in the file itself, but only in derivatives of the file. The definitions or substitutions can be called out, when needed, from disk storage.

The uses to which these notations can be put are unimaginably versatile and complex. It is possible for globals to be substituted for globals. For example, one global notation can be redefined as several notations, and then each of these can be treated differently. Changes in format can be effected, but so can changes in text. Interpretations can be injected for the convenience of the proofreader, or for the instruction and reassurance of the customer. These can be removed when they no longer serve their purpose. Such notations can help identify critical passages for indexing and abstracting, or for compiling lists of authors and titles or other relevant data. When used for identification they can also be used to impute to the material so identified whatever format and typographic characteristics one may, at that moment, desire. Nothing is, of course, irrevocable.

The global notation is somewhat different from the global editing concept which we shall discuss in Chapter XIX, but there are similarities. The global notation is a purposeful combination of unique characters susceptible of infinite redefinition, but of themselves signifying nothing. The global correction is the redefinition of a word or phrase or combination thereof, or any other combination of text and instructions which in and of itself has precise meaning. Obviously, these two concepts are interrelated and in some situations may be used interchangeably.

Global notations emerged from experimentation with keyboard shorthand techniques, and also out of the need to generalize into undefined format statements very specific typographic commands. The use of such notations is but an evolutionary step in the direction of highly sophisticated applications which will enhance and dramatically magnify the capabilities of the computer typesetting system as opposed to conventional methods of composition.

**Review Questions**

1. Give an example of the use of keyboard shorthand to simplify the inputting of text. Differentiate between global situations and self-cancelling shorthand conventions.
2. How does a format statement compare with the concept of global notation? What can the latter do that the former cannot do?
3. What is the difference between a global notation and a global correction or global substitution in an editing pass?
4. At what point of time, in the processing of a manuscript, should the global substitution take place?
5. What are the arguments for a permanent substitution of an intentional string of characters for the abstract global notation, as opposed to keeping the abstract string in the file?

CHAPTER XVII

# The Typesetting—Data Processing Relationship

Of course any computer manipulation for typesetting, such as hyphenation and justification, or pagination, is a form of data processing, but in the typesetting field a distinction is generally drawn between *typesetting applications* and *data processing applications.* The first category relates to one-time typesetting (where the product is composed once and the file will never be re-used) and also to reiterative typesetting (where a file is created so that it may provide more than one setting of the same material, although perhaps in different formats, but there is no intention to use the file for purposes other than typesetting, nor has the file been created for any purpose other than typesetting).

The second category relates to the *creation of a file for purposes other than typesetting.* Typesetting may be a by-product, with some other use of the data serving as a principal product. Or, typesetting may be the principal purpose, but the file also serves other purposes. Or, even though the objective is typesetting of all products, the products themselves are created by means of computer programs which perform functions other than those required for typesetting. These are data processing functions, for, as we use the term, such functions take place only because they are necessary in order to create new or different products.

And so in this chapter we shall be concerned with the manipulation of data for purposes other than typesetting, and with the interface between these two activities. By interface we mean the problem of moving back and forth from one application to another.

### The Biggest, Most Profitable Market

At this point of time there seems to be general agreement that the most profitable opportunity for computer typesetting lies in these data processing applications. It is felt that "straight typesetting" by computer is not likely to result in costs low enough to compete with conventional means of setting type. Hence it is logical to suppose that those uses which somehow tie in with data processing will represent the major area of endeavor.

What may be true today may not be true tomorrow. Certainly, smaller computers and inexpensive photounits are more likely to be engaged in straight typesetting activity, and they represent the overwhelming number of installations, even today. But the larger computers, supported by high-speed CRT output devices, are of course capable of creating fantastic quantities of type, and it is these larger installations which so far have not generally found it to be advantageous to concentrate on typesetting for its

own sake. This situation may change. On the other hand, we will find many more uses than mere typesetting for the data that was once available only in this form.

It is extremely important, therefore, to understand how typesetting and data processing may be brought closer together to their mutual advantage.

### Types of Markets or Uses

Suppose we look at the subject from the point of view of the computer typesetting supplier in search of markets or applications for his equipment and software. What uses will he find?

### Substituting for the Line Printer

It is believed that there are many occasions when it is desirable to typeset what would otherwise be presented as computer line printer output. If more than five or six copies are required some superior form of replication is necessary. Printouts can be photographed and reduced to printing plates, but type will often do a better job. Type is proportional and takes up less room. Type lends emphasis and clarity, because varying sizes, weights and faces can be employed. If a thousand or more copies are to be printed the paper savings alone will warrant going to type, provided it is not necessary to re-set the computer output—in other words, if the information in the computer can be reformatted and processed through a typesetting program. Then typeset output can be obtained.

Such reformatting can be performed by the computer typesetter or by the customer who generates the line printer output. If it is done by the customer, the typesetter only gains in that he will derive revenue from the sale of photounit time. He will receive no revenue from the use of his computer or his programs. But photounit time is what he has the most of, and he is generally delighted to find someone who is prepared to "hang" a tape on his CRT device for a relatively long run, and who will be a source of very little trouble, demanding nothing of keyboarders, proofreaders or programmers. However, few customers have the programs and the capability to provide output in a form which can be accepted directly by the photounit. The ones who have developed this capability are customers whose volume is so substantial, and the economy offered by this method is so great, that they have taken the time and trouble to create the special-purpose interface which their jobs require.

### Programming the Customer's Data

Another alternative is for the typesetter to "massage" the customer's data, formatting it for the photounit. This can be done either by a generalized program which interfaces with the supplier's typesetting software, or a special-purpose program which performs whatever functions are necessary, bypassing the composition program, but preparing the data to be handled by the photounit. If the definition of the assignment is to set in type, line-for-line, what would otherwise have been composed on a line printer, the task is generally not a difficult one. Hyphenation and justification are not needed, and page makeup will be the same as that which previously had been performed on the line printer. A one-time program will handle most jobs of this kind.

It often happens, however, that the customer possesses a data file he does not understand, since it was developed for him by another supplier, engaged in conventional data processing activity, and he wishes the typesetter to pull from that file only certain data elements, and to reformat and perhaps resequence these elements. The

typesetter will necessarily have to develop a special program to handle this one task, although he may have created certain subroutines which he can string together quite simply; or he may find it expedient to prepare the program in an appropriate compiler language, such as PL-1. After selecting and manipulating the data, the typesetter may then format the run so that it will pass through his composition program, or he may decide to bypass it altogether.

Perhaps there is no special reason why the typesetter, rather than the customer's usual computer center, should find himself deeply immersed in data processing activity of this character. On the other hand, he, more than anyone else, knows the input or format requirements of his composition programs, or, alternately, of his typesetting output device. Moreover, he will have developed a certain facility for character-by-character processing which is somewhat foreign to other computer users, and he should know more about his customer's literary and stylistic requirements. For example, conventional sorting or sequencing runs, taken off the shelf from a computer center's supply of utility programs, will often not cope with the somewhat more elaborate rules the customer has developed over the years. Library catalog sequencing, for instance, is highly specialized, and even the mundane telephone book presents some interesting problems; moreover, these seem to differ from company to company even within the Bell System.

The opportunity thus presents itself for the computer typesetting establishment to become a specialist in the handling of data for literary applications. These involve many peculiarities not otherwise encountered in the computer world. Points and picas, extended character sets, widows, telltales, principles of indexing, the ability to arrange and present material economically and artistically—these should be the stock in trade of the new compositor.

### General Categories of Activity

The listing below may serve to indicate the variety of ways in which typesetting and data processing activity can become intertwined:

1. The typesetter merely provides the output device, accepting input from another computer.
2. The typesetter accepts as input a tape formatted for the line printer, and processes it through a special program which prepares a tape for his output device.
3. The typesetter accepts as input a tape containing data in the proper sequence, with the necessary generalized codes imbedded at appropriate places in the text stream. He reformats the tape, translates or interprets the codes, and processes the material through his composition program.
4. The typesetter accepts as input a tape which contains only the data to be typeset, but which requires all codes to be inserted by program. He prepares the program and processes the tape through his composition system.
5. The same as point four above except that he develops a special-purpose composition program, due to the size of the job or its repetitive nature. The special program will run more economically.

6. The tape supplied to the typesetter contains irrelevant information. The typesetter pulls only the data he requires, and then proceeds as in points four or five.
7. The same as point six except that the typesetter resequences the data he selects from the supplied input.
8. The typesetter creates the original input for typesetting, but pulls from his file and returns to the customer data for use in other computer manipulations.
9. The typesetter creates the original file for typesetting and for other purposes. It is a structured file, which permits him to perform such operations as *extraction, explosion, sorting, abridging, abstracting,* and *indexing.*
10. The file created by the typesetter is heavily encoded, but is not structured. It consists merely of a running stream of text. Nevertheless, he is able to extract certain data from the file, manipulate it, and either feed the results back into the composition program, or generate other non-typesetting by-products, such as mailing lists or computer printouts.

Obviously this listing is only illustrative, and there are many combinations and permutations of the points enumerated. For example, from a text-stream file the typesetter may generate an index for an unabridged edition. He may then create an abridged version and generate an index from this. Or he may create an index each month from a certain publication and then cumulate the indexes quarterly, annually, and so on. He may even cumulate the articles in the publication, especially if they are brief abstracts which are most conveniently gathered together in a more logical fashion than the manner in which they first appeared.

Moreover, in all of the examples listed above, we have assumed that when data is supplied to the typesetter it will first have been processed by another computer. It is entirely possible, on the other hand, that the typesetter is provided with initial input which may or may not have served some other purpose. For example, his customer may have a price list which he maintains on punched cards. He now wishes to typeset this list. Formerly he produced a printout from his unit record equipment. (He had, let us suppose, a punched card system which antedated the computer.) Or, the customer has keyed his data directly onto a magnetic tape by some sort of key-to-tape device.

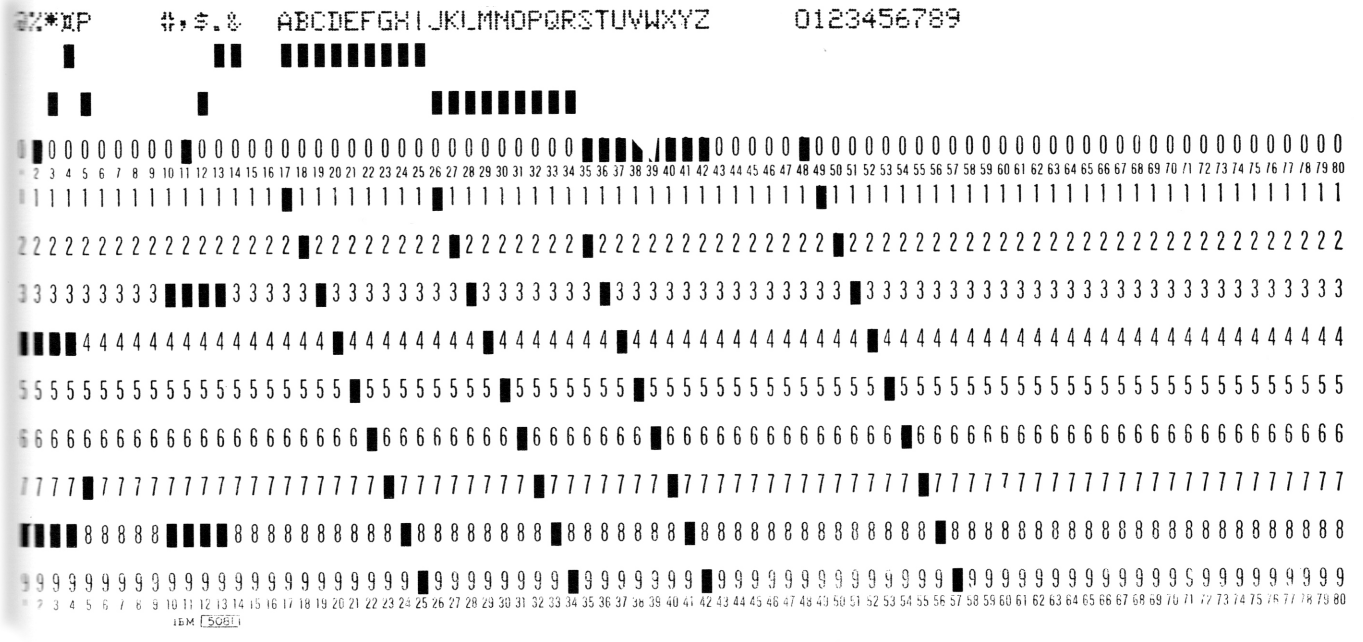
### Characteristics of Data Processing Activity

Probably the single most relevant characteristic of conventional data processing is the existence of *structure* to the data. In other words, the information is *fielded,* and the most usual arrangement is for the data to be structured in *fixed fields.*

We shall assume the reader knows nothing about data processing. The best way to begin his education is to consider the punched or Hollerith card, since this is the way in which most structured data (and some unstructured input as well) comes into being in the data processing world. The usual punched card contains 80 columns. These columns may be arbitrarily divided into fields. In each column one character may be written. Twelve positions are available in each vertical column. The ten bottom rows represent numbers, 0 through 9. One punch hole per column is thus used to indicate figures. When a hole is also punched in one of the top two rows (or in the zero row)

and *also* in one of the rows 1-9, a non-numeric character is indicated. For example, *A* consists of a hole in the top row and a hole in the *1* row. *B* is made up by a hole in the top row and a hole in the *2* row. The letter *J* is represented by a hole in the row which is second from the top and a hole in the *1* row. *S* is signalled by holes in the *0* and in the *1* rows. A total of 41 characters can thus be described, consisting of ten numbers, 26 capital letters, and five miscellaneous characters, some of which serve as control codes, but the period, comma, semi-colon and dollar sign are frequently used. It is possible to use the same card to convey more information by punching more holes in these vertical rows, but this is seldom done. However, the use of an upper and lower case alphabet is sometimes encountered.

Two significant factors emerge. One is that the character repertoire is extremely limited. The other is that the position of the data in fixed fields, across the card, enables the program (either the wired circuit board of a unit record equipment, or the software of the computer) to identify each element.




## A Case Study

Let us suppose, for example, that we wish to prepare a listing for a college alumni register, and we wish to present the following information: name, address, class, and department of specialization. We want to be able to produce one listing in alphabetical order irrespective of class and department. Another listing is desired alphabetically by year of graduation. A geographic listing is also sought, again arranged alphabetically, but by city and state. Finally, perhaps a listing by department or specialization, irrespective of geographical considerations, may be useful. For some listings only the name and class will appear (not the individual's address). We will keep our example simple by presuming a male college, so that women's married names do not complicate the picture, and we will not differentiate between the dropouts and those who were graduated.

Since we are going to want to sort on names, we must consider how this is done. We must have a field designated for the last name, and another for first and second names. (If there are "Jr.'s" or a presumptuous "III," this will become part of the given name rather than of the surname). How long should the field be for the last name? If it is not long enough it would unfortunately be necessary to "truncate" someone's name and, if the list is ever to be published, alienate him and lose his contribution to the alumni fund. If we had all names on a computer we could search for the longest name. But we are just starting to create the file. So we ask around and try to learn what others have done. We also scan the printed lists available to us. We finally determine that the longest name of a graduate was Allenby-Whittall (and now we know we need a hyphen in our character repertoire!) but we decide to allocate 18 columns for the last name field. Although we keypunch only the characters required, from left to right, our program will fill up the remainder of the field with "trailing zeros" in order to create a uniform sort field:

```
AARON0000000000000
AARON0000000000000
ABBOTT000000000000
ALLENBY-WHITTALL00
BEAUCHAMP000000000
```

and so forth.

We decide, then, to allocate twelve columns for the rest of the name. Hence the first name will begin in the nineteenth column:

```
AARON0000000000000JONATHAN R00
AARON0000000000000LEONARD S JR
ABBOTT000000000000ARTHUR W III
ALLENBY-WHITTALL00PERCIVAL0000
```

and so we are well started on our "system" design.

Now, turning to the problem of addresses, we visualize that we will want to arrange our information by state and city. What do we do about suburbs? Should "Shaker Heights" be listed as a part of "Cleveland"? Perhaps we are better off sorting by zip code, and then playing the tape against a zip code identification file supplied by the Government. But let us ignore this possibility and try to solve the problem ourselves, and we shall take the lazy way out by deciding to sequence the geographic file by state and then by town, without any effort to group adjacent towns. Hence we decide to "field" the address portion of the entry as follows:

House number and street
Town
State
Zip code

Let us consider the problem of *state* first: the easiest way to represent the state is by a two-digit alpha field: *MA* for Massachusetts, *CA* for California, *NJ* for New Jersey,

etc. However, if we are going to sort on these abbreviations we will get our states out of order. *NC* for North Carolina will sort ahead of *NJ* for New Jersey, for example. Or, we could number the states sequentially. If the file is only to be used for typesetting these numbers could be stored in a "table look-up" and the name of the state could be generated. We can do the same thing for the computer printout run for mailing labels. It is only if we want a card print that the initials or abbreviations need to be used. On the other hand, it is easier to use the alpha abbreviations than to have the input operator perform a look-up operation in order to assign state code numbers. So we settle for a two-digit alpha representation and we assign to our computer program the task of sorting these in the proper order at a later stage of the game.

So far as town or city goes, we can truncate the town if we like, in case it is too long, and again use a computer program to generate the full and complete name, if this is desirable for reasons of clarity or taste. It might reduce keyboarding if we told the operator to input only the consonants and use a table look-up for all names. On the other hand, it might result in creating some redundancies. Moreover, a listing of many hundreds of towns would be necessary in order to interpret the input mnemonics. So we conclude we had better keyboard the town's entire name to the extent that it can be represented by fifteen columns. We tell the keypunch operator that if it cannot be so represented, to truncate at 15, and write down these names so that we can expand them in our computer program. This will reduce appreciably the number of exceptions we shall have to contend with.

We do not anticipate having the need to do anything special with the house number and street field, other than to print or typeset it, but we must allow adequate room for it.

Three other fields remain. We must designate the graduating class for each person listed, and we can do this by two numbers, such as *60*. (We can assume that all numbers referring to years yet to come actually represent those years in the preceding century (such as 1895). The next field will indicate the major subject of the alumnus. One digit will suffice, since we are pressed for space, since we can indicate 36 different possibilities by using a letter or a number in this column. Finally we wish to indicate whether the degree conferred is an A.B., a B.S., or perhaps a Ph.D., D. Litt., or M.D. Each degree can be given a number, and *1* may as well stand for the A.B.

By careful planning we have just reached the allowable 80 columns, and we can avoid the necessity of having to overflow onto a second card. Here are the allocations:

| | |
|---|---|
| 1-18 | last name |
| 19-30 | first name |
| 31-54 | street address |
| 55-69 | city |
| 70-71 | state |
| 72-76 | zip code |
| 77-78 | class |
| 79 | field of specialization |
| 80 | degree |

It will now be possible for the punched card itself to serve as input to a print format program which would generate labels. Additional programs can provide geographic and class listings, or could produce a sort by kind of degree or major subject.

The master file can be kept on punched cards, and when addresses are changed the cards can be physically removed and new ones put into place. Or the file can be maintained on magnetic tape or disk, in which case a file update program will need to be written or adapted.

Our typical entry will read as follows:

```
AARON0000000000000JONATHAN R00APT C 17698 MACARTHUR AV0
WINSTON SALEM00NC6031
```

### Going into Type

Let us suppose, now, that we are given an assignment of typesetting the alumni directory from these punched cards, or from a tape image of the cards. The first listing we are to produce is to be the master list in the alphabetic sequence in which the file is maintained. Our program would work something like this:

1. Read in a dummy entry which will set up the job, defining the line length and giving the program other data it may require.
2. Read in the first regular entry and store it in core.
3. Take the first field and move it to a new location, dropping the trailing zeros, and inserting, in front of it, a flag which will subsequently identify the type face. Follow the field with a comma.
4. Read in the second field, dropping trailing zeros, insert another typographic flag if the type face is to be different. Follow the field with a period.
5. Then go to positions 77 and 78, put a flag ahead to indicate the desired type face, then an apostrophe (as in '90) and follow the two digits with a comma.
6. Then go to position 80. Do a compare or table look-up which indicates that because the value in this column is *1* the letters *Econ* should be substituted, with a flag to designate italic. Follow with a quadleft code.
7. Then go back to position 31 and pick up the string from 31 to 54, dropping any trailing zeros, and preceding the entry by an em space, to cause the address to indent. A typographic flag code may also be added. Note there is no punctuation in this field. We may wish to interrogate this field for certain peculiar conditions. We will return to this point later.
8. Then follow directly with the name of the city. Perhaps this name has been truncated. (We hope not!) But we may require a routine to check first to see if there are any trailing zeros in the field. If not, the name may have been truncated. The name would then have to be compared against a list of truncations, and if found, then the full string of characters spelling the entire name would be picked up. Preceding the city name would be a flag for the desired type face, if different. A comma follows the end of the field.

9. Then proceed to columns 70 and 71. Let's make it easy and assume we are merely going to pick up the two letters we find there. We recognize that we have not placed a period after the *N* in *NJ*, and so presumably we decide not to put a period after the J or after CA, so we insert a comma, and follow with field 72-76, where we pick up the zip code, and end with a quad left, and then a blank line before going to the next entry.
10. We write this information to magnetic tape or disc, and it is now, hopefully, in the format required to enter our composition program. Perhaps we have also inserted other flags to indicate the beginning and ending of an entry so that our pagination program will not break entries when it divides the material into columns and pages.

Although this description seems complicated, the task has been a relatively simple one. But since we also are to be required to produce a listing by state and town we might have given consideration to creating the information we needed for this purpose at the same time that we prepared the composition input tape for our main listing. In the interest of clarity, however, let us assume that we regard this as a separate and distinct processing run.

We return, then, to the initial file, and we recognize that we must create a sort field. First we wish to sort by state, and then by town. We know that the number of states (fortunately there are no foreign countries!) is less than 99. We call in the state field and assign a number. A translate table tells us that CA is California, and that California is the fifth state in the alphabetic sequence. And so we write out *05*. We follow this with the name of the town, allowing fifteen places, and then we pick up the last name of the alumnus, since we also wish to sort alphabetically by last name where there are several persons listed as residing in the same city. We must also consider the possibility that two persons with the same last name will live in the same town, and we need to pick up, as well, the first name of the alumnus. And so we have created a sort field as follows:

```
05WHITTIER0000000ALLENBY-WHITTALL00PERCIVAL0000
```

In addition to creating this sort field, we will need to develop a "trailer" field which will contain the information we wish to typeset. In this case all we wish is the name and class of the individual. Let us suppose, although this is unlikely, that we decide to set the first name first, so that under California, and under Whittier, we will find:

PERCIVAL ALLENBY-WHITTALL, '59

There is no problem in doing this. The sequencing will be by last name, but the listing will begin with the first name. The only difficulty we will encounter is that we may have instances like this:

LEONARD S JR AARON
ARTHUR W III ABBOTT

because we did not take the trouble to "field" these post-surname characteristics. (If we had we would not have been able to "run in" this information in the same field, and more space on the fixed field card would have been required.)

At any event, after we had sorted our entries by the use of the sort field, the program would drop it and pick up only the "trailer" which contains those fields of typesetting information we require, in the sequence in which they have been arranged, and with the appropriate "before and after" information around them. By "before and after" information we mean such things as flags to indicate type faces or format instructions, and repetitive information, such as commas or dashes to separate one element from another, as well as to separate one entry from another.

That is all there is to it. One thing should be clear: it is easier to develop such a program if there has been communication from the very beginning about the typesetting possibilities of the input, rather than to try to improvise from a file which was created without such uses in mind.

Consider, for example, the problem of setting the same file in upper and lower case. The customer proposes that this be done, since it will be more legible and will take up less space. An investigation of the name field reveals certain peculiarities, however. The solution is not so simple as to pull off the first letter of the last name, make it a capital, and then to convert all of the other letters in that last name to lower case. We encounter *McAdoo* and *MacDonald,* but we also find *Mackey.* There is a *Dupuy* in the class of '42 and a *duPont* in the class of '98. The latter, by the way, is a substantial contributor! Some of these problems can be solved by special routines, but now we are obliged to carry on a different level of data processing. Until now all we have been concerned with is to place characters at the beginning or the ending of identifiable fields (or in some cases to translate a field from one thing to another). Now, unfortunately, we have to make an analysis of the contents of the field itself. We can write a program to compare for *Mc* and can probably get away with the capitalization of the next letter. We can gamble with *Mac,* although the consequences will be somewhat more treacherous. We'd better give up on the *du* and *de* problem. An editing pass may be necessary to get the file into shape for typesetting. But if we do this, can the file be used again for other purposes, such as the generation of a mailing list? Or will it be necessary to edit the file each time a typesetting application is planned? These are difficult system questions to answer. Again, advanced communication is indicated. If a dollar sign could have been placed ahead of the exceptions the problem could have been resolved nicely:

```
MAC$DONALD
$DU$PONT
```

which tells us to reverse the normal capitalization rule, but this takes up more room in our fixed field areas, and these irrelevant marks may interfere with the use of the file for mailing lists or for sorting, unless we incorporate into these programs some procedure which causes the routine to ignore them.

If it is difficult to infer the proper position of upper case letters in the name field, it is all the more perplexing in the address field. The rule that the first letter in the word should be capitalized would produce *Afb* rather than *AFB* (for Air Force Base) for example!

### Creating a Typesetting File

Just as he often works with fixed field data supplied to him by the customer, so our computer typesetter may be asked to assist in the development of the initial file, and

he will possibly choose to use some other method of "fielding" the information. He may elect to use variable length fields, and he may decide to place these in random order, or even merely as a part of a text stream. But let us take a case where the sequence of elements would be prescribed. Suppose, for example, he wishes to create a library catalog. He may develop a concept which permits the entry to look this:

```
$aCooper$bJames Fenimore$cThe last of the Mohicans$dMacmillan$e
1896, 395p.$g$h$i$j$k$z
```

We have flagged the author's last name between *$a* and *$b*. We have flagged his given names between *$b* and *$c*. Thus we can find these elements and create sort fields when necessary. We can also find them when we wish to pull them out for typesetting.

The title of the book appears between *$c* and *$d*, and a program can be written to drop off the articles (*The, A, An*) when a title sort is indicated, in which case the next work should begin with a capital if it does not already do so.

The publisher appears between *$d* and *$e*. Miscellaneous information which we shall typeset but not use for sorting, and which, when typeset, will always appear together, is found between *$e* and *$f*. The identification of the category of book appears between *$g* and *$h*, and in this instance the *F* stands for fiction. Thus we can sort all fiction together, or identify the book for the reader by Dewey number or LC number, and sequence on these fields with a little legerdemain. Finally, we have indicated that this book can be found in three libraries in the system, which we have chosen to identify uniquely by the codes *$i, $j* and *$k*. The entry ends with the flag *$z*.

This is a simple application. Library catalogs are of course much more complicated than this, since they also deal with subject indexes, but the example suffices to make our point. It is now necessary to have a program which responds to commands to create sort fields and trailer fields, generates information before and after the fields in the trailer, and otherwise manipulates the entry much as in case of the preceding illustration (from the alumni directory). But since the file was initially input for typesetting, upper and lower case codes, and other internal flags, to extend the character repertoire, etc., can be widely used, so long as they do not interfere with the sorting rules. Sorting rules themselves can be made more sophisticated by intermixing "file as if" commands where normal sorting sequences do not produce desired results.

### Other Opportunities and Problems

For sorting purposes variable length fields must be converted into fixed fields, so that apples can be compared with apples. But for other purposes variable length or random fields are much more convenient. It is not necessary to structure the input in a rigid fashion.

In fact, we might consider this type of input to be non-structured. The concept of "structure" in this sense would mean that the item would be identified by its sequence or its position or both. Naturally, there can be some element of structure even in a random file, and wherever structure exists certain conditions can be imputed to the data, so long as a beginning and ending can be determined.

Flags can be put around names which occur in random order in the text stream so that these names can be recognized for the purpose of creating an index. Material to be omitted for an abridged version can also be identified. First lines of poetry can be tagged so that an index of first lines can be generated. These are only a few illustrations which come to mind. Space does not permit a further elaboration of this topic,

which is obviously deserving of an entire book in its own right. But perhaps enough has been said to start the reader thinking about the ways in which typesetting and data processing can be interrelated. Since publishing need no longer be considered exclusively a function of typesetting and printing, it is obvious that many publishers will wish to store their products in such a way that they can provide information through a number of media, including a magnetic tape file which others can search or manipulate for their own purposes. Information retrieval and the selective dissemination of information is also possible from files which have been input with these applications in mind. The more thought that goes into the development of the original input the greater the potential for byproducts and alternate uses.

### Definition of Data Manipulation Terms

Some forms of data manipulation often used in typesetting applications include:

Extraction—selecting only certain elements from the text, such as the name of an author, or a citation or title.

Explosion—generating the same information in different arrangements from the same initial input. For example, the name of a company followed by the products it produces, and then creating a list of products with the company name restated under each product.

Abridging—eliminating certain material from the text stream.

Abstracting—summarizing the essence of an entry or article.

Automatic indexing—generating a listing of index terms in alphabetic sequence, with reference to the page numbers where the item occurs. The index terms are selected by computer program, such as to pick out all words in the text stream of five letters or more in length, and assuming that words so used with frequency represent key words in the text.

Redundancy elimination—dropping a portion of an entry when it already appears in a preceding context. For example:

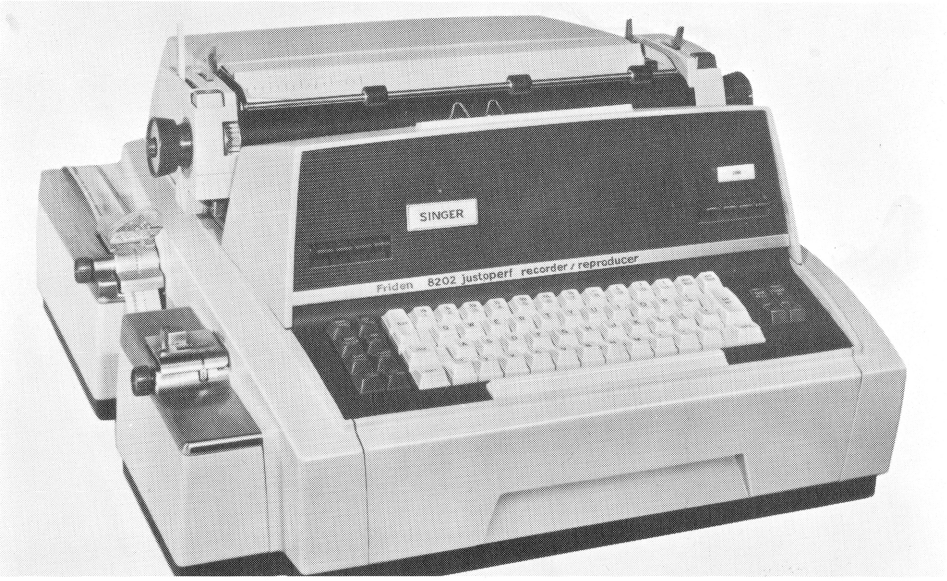
U.S. History—Colonial Period—Early Settlers, p. 67
—Handicrafts, p. 14
—Revolutionary War—Anecdotes and Satires, p. 4

### Programming Implications

While it is true that the use of the file for other purposes makes computer typesetting much more viable, economically, (and everyone these days wants to find himself a unique data bank which he can exploit on the market place!) programming and systems design are complex and far from trouble-free. Computer typesetters who have forsaken straight one-time setting for data bank applications, on the ground that they cannot make a profit in competition with conventional typesetting, also find that the data processing path is a thorny one. Programming ingenuity is obviously necessary, plus the ability to anticipate the customer's requirements and point out the pitfalls before they are encountered. These skills are rare, and their absence keeps the typesetter from profiting by the opportunities he is able to uncover. Moreover, neither he nor the publisher may possess the insight to discover the wealth of applications which lie around them.

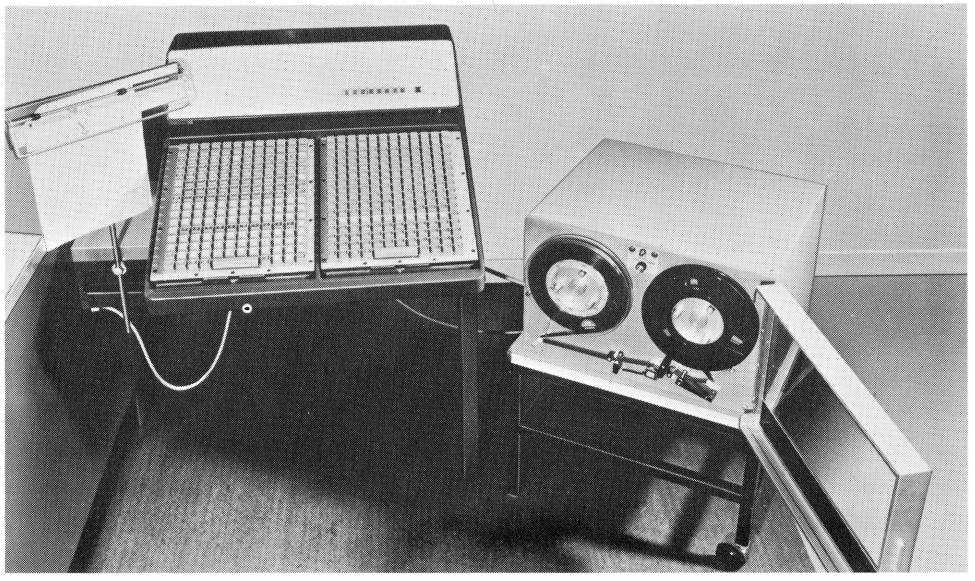
**Review Questions**

1. What is the "interface" between data processing and typesetting?
2. What is meant by the typesetting of line printer output?
3. What are the arguments for general-purpose data processing programs as opposed to special-purpose ones, and vice versa?
4. List various ways in which a computer typesetting establishment may get involved in data processing activity.
5. What are fixed fields? Variable length fields? Why are "flags" sometimes needed?
6. Explain how the Hollerith punched card may be read.
7. What is a sort field? A trailer field?
8. Why should the elements within a sort field be the same length?
9. Explain "explosion," "redundancy elimination."




THE 8202 JUSTOPERF* TAPE PERFORATOR FROM SINGER'S FRIDEN DIVISION IS AN ENDLESS TAPE PERFORATOR THAT PRODUCES HARD COPY. *A TRADEMARK OF THE SINGER COMPANY

THE FAIRCHILD ELECTRO/SET IS A PRODUCE OF FAIRCHILD GRAPHIC EQUIPMENT COMPANY.



LIKE THE KEYMATIC, THE MONOTYPE MODEL MT PROVIDES A WIDE RANGE OF KEYS FOR MIXING AND OTHER SPECIAL PURPOSES.

CHAPTER XVIII

# Preparing Text Input For Computer Composition

Text input will be assumed to consist, basically, of unstructured data, not "fielded" in either fixed or variable length records. The text stream or "intelligence" of the document will need to contain certain codes or flags, however. These may be included in the initial input, they may be added later, or some of them may be included initially while others are contributed during an editing process.

The codes or flags in question may be said to be of the following kinds:

a.) flags which are designed to enlarge the normal character repertoire.

b.) flags which label the various type faces or sizes in which the manuscript might conceivably be set. We shall call these *font designators,* considering a font to be a unique set of graphics of a given size.

c.) flags which identify elements which may require the application of special rules of format (irrespective of paging requirements).

d.) flags which identify elements which need to be considered in connection with page makeup.

e.) special flags which may serve other miscellaneous purposes, such as the identification of names to be extracted for an index, material to be deleted for an abridgment, or data to be embedded in the text for reasons other than typesetting.

It should be evident that some of the flagging functions can overlap—that some flags can serve more than one function, or that some flags can be redefined so that on one occasion they serve one function and on another they serve another. For example, a flag can identify a head as a "number one head" for purposes of defining a type style (as in point *b* above) and for purposes of identifying a format for that head (as in point *c*), and for the purpose of identifying a condition for page makeup—namely, do not break directly after this heading (as in point *d*). Moreover, all number one heads can be identified for extraction in a table of contents (as in point *e*).

### Extended Character Sets

Let us consider first the flags necessary to define the use of extended character sets. We have already mentioned the need for the development of conventions for the grouping of characters or "sign subsets." We shall now propose a set of conventions designed to meet most conditions. We are doing this for two reasons: first, because we know of no other adequate way to identify the problem; second, because it may be that our solution may be found to be generally acceptable. Readers may wish to adopt these groupings and flags, since they have been developed with thought and with the benefit of considerable knowledge and experience in the field of book composition. But if they feel that they can do a better job—and they certainly can do so for any given special-purpose application—they may nevertheless profit by our efforts in this connection.

We have endeavored, for reasons that we have discussed elsewhere, and especially in our chapter on "Input Theory" (Chapter XV) to create character sets, or subsets of signs, that can be described within the range of characters available on a conventional typewriter keyboard, even if this is not the ideal device for the creation of input. More "options" are preserved in this fashion, and larger groupings, on special-purpose keyboards, can consist of more than one such subset.

The reader is warned at the very outset that what we are going to show here is complicated. We have made up our mind that there should be no compromise with the basic principle that it should be possible to keyboard any character that will be required in book composition, and there are of course thousands of sign-graphic-size-position combinations. We visualize, for example, 25 flags for fonts, where fonts are defined as unique graphics of a given size and design. These fonts might consist of text roman, text italic, text bold, extract roman, extract italic, extract bold, a similar grouping of three fonts for bibliographic listings at the end of the chapter, and another grouping for footnotes. There may be several levels of display heads, plus credit lines, plus legends, captions and the like.

We also visualize, within these 25 fonts, the possibility of ten (or more) different character sets or sign subsets.

Finally, we visualize, for any one of these 25 x 10 combinations, the possibility of positioning these characters as superiors or inferiors, or superiors-to-superiors and inferiors-to-inferiors. Thus we are literally describing 1000 possible arrangements of sets of 80 characters, or 80,000 different graphics. Moreover, since we make extended use of floating accents, and there are other typographic tricks that can be played, the system we describe could, in all probability, cope with over 100,000 different combinations. This does not include the fact that these 25 basic fonts can be redefined to consist of other type designs, so that Baskerville can be substituted for Times Roman, or Helvetica for Univers.

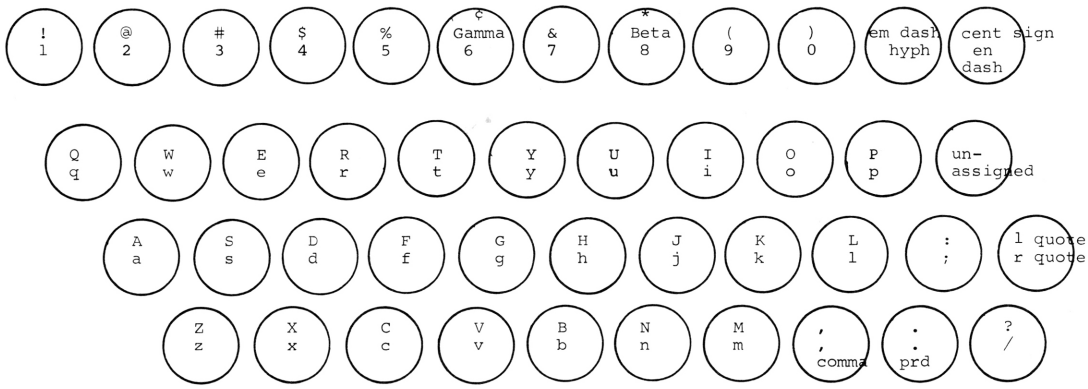
Although our system copes with foreign accents and can set Greek, Russian or Hebrew (or any other language) it is English-language oriented, and it has been devised so that the program would know which characters are "alpha" characters that may be submitted to a hyphenation routine, and which are not. Basically, only the alpha characters in keyboard layouts 1 and 2 would be admissible for hyphenation, but accents, provided in layout 3, will not interfere with the division of the accented word.

There are 44 keys on a standard typewriter. Considering the upper case or shift feature, this allows for 88 characters. But our system provides for groupings of 80. From the 88 we subtract two flags, which we shall call Beta and Gamma. We also provide for one key (two positions) to be defined optimally and optionally for any given job, by an initial prephase, and thus avoid the need for flagging special characters frequently used. And two keys we reserve at all times for the em dash, hyphen and en dash, in effect removing them from the "standard" layout because they appear on all layouts. Of course, certain other characters are common to several layouts.

The spacebar is universal, as the justifying space. The carriage return can be defined either as a justifying space or as any unique code or code string desired. Em, en and thin spaces (and other fixed space codes) must be described by key combinations on this limited keyboard, and this is also true of all flags except the Beta and the Gamma, which are themselves first characters in flagging sequences.

Hence the Beta (*) and the Gamma (¢) are universal flags. Two are needed, as we explained in our chapter on input theory, to simplify the input keyboarding on a regular typewriter. We selected these two key locations because they can be struck with the stronger fingers. An alternate graphic representation might be desirable, and those with Selectric typewriters might wish to designate such symbols as and for Beta and Gamma respectively.

Here is a diagram of the standard assignment of signs which we shall call layout #1.




Possible Standard #1 Layout for Sign Subset

As noted, we have given up the asterisk. If we keep the cent sign on the layout for use as a cent sign, we can move it above the en dash.

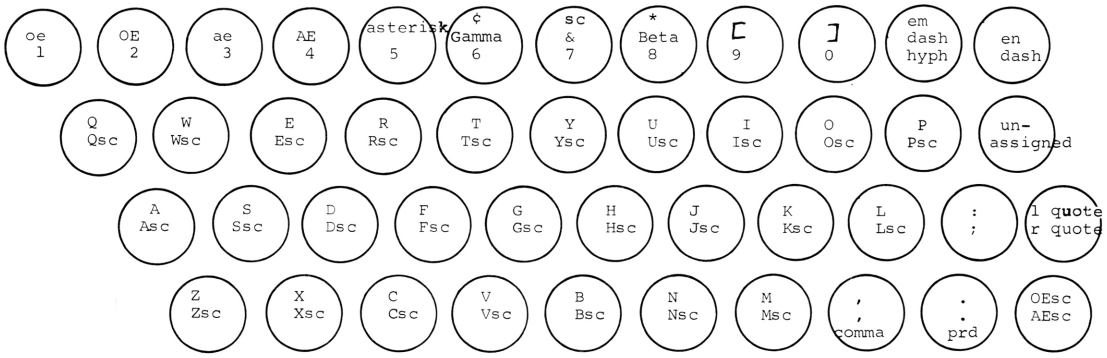
On this particular typewriter the hyphen and the underscore will become the hyphen and the em dash, the equal sign becomes the en dash, and the "plus" is the cent sign.

| | |
|---|---|
| - | hyphen |
| _ | em dash |
| = | en dash |
| + | cent sign |

One unassigned key is shown, with the thought in mind that the two positions on this key could be non-standard, and a message at the beginning of the job would give the job planner the opportunity to assign two special characters in common use within that job, and thus avoid the need to switch to another subset.

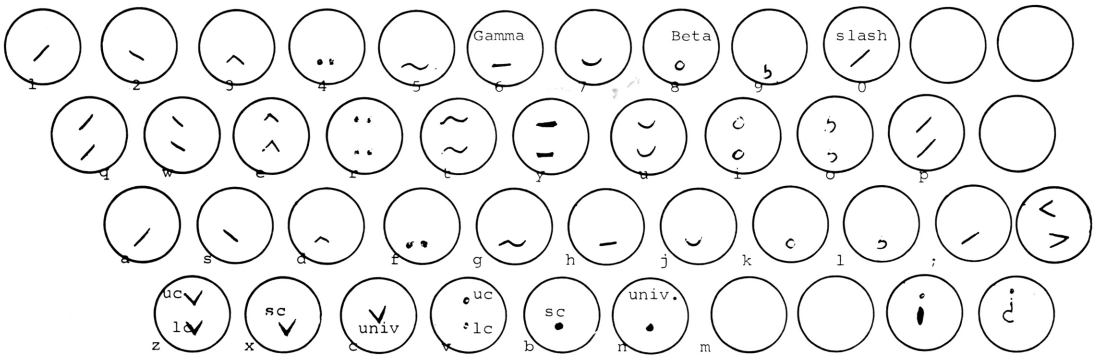
Keyboard layout #2 is designed for caps and small caps. A job can be defined in such a way that the alpha characters can be treated either as cap and small cap or as cap and lower case. Whereas ligatures will be program-generated, since they do not always occur in all fonts, diphthongs must be keyed uniquely, since they do *not* always

occur where these letter combinations exist and where diphthongs exist in the font. A special routine would be required to submit such words to the hyphenation module, and this routine would only be invoked if layout #2 is used.




Possible Standard #2 Layout for Sign Subset
including small caps and dipthongs
Figures can be Old Style (non-lining)

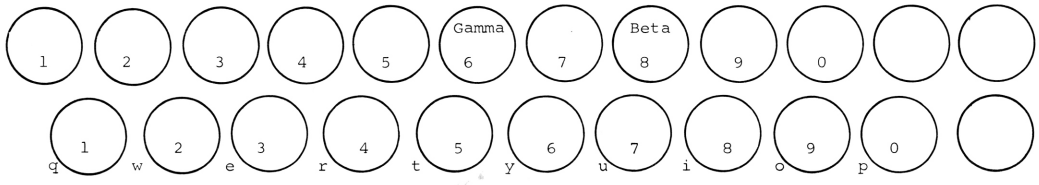
Various options are provided in keyboard layout #3. If the first row of accents is used, plus the two "universal" accents in the bottom row, the program will be expected to determine whether the accents are to be positioned over caps, lower case letters or small caps. If the program does not carry this feature, then the second row accents would be used (upshift for caps, downshift for lower case) plus the third row for small caps, and a few spill-overs on the bottom row.




Possible Standard #3 Layout for Sign Subset
Top row: universal accents if program-generated.
Second row: upper and lower case accents if not program-generated.
Third row: small cap accents if not program-generated.
Bottom row: extra accents.
Accents are Non-Locking.

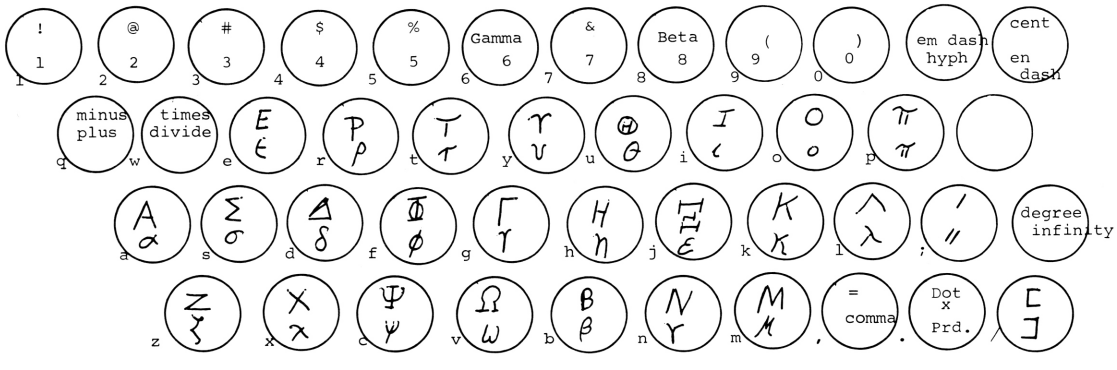
As noted, a few other characters which are needed for foreign language setting are also included in this layout, such as the upside-down exclamation point, the question mark, and the French quotes.

Fractions can be generated by using keyboard layout #4. Parameters will specify whether the fractions are to be "stand-up" or created with the use of the diagonal slash.



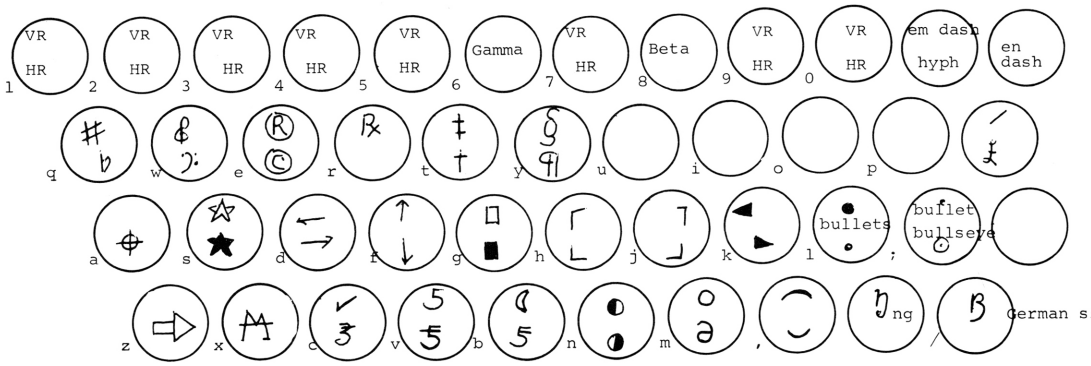

Possible Standard #4 Layout for Sign Subset
Top row: numerator fractions
Second row: denominator fractions
For slash use extreme key, bottom right.
Program parameters will define whether slash is
horizontal or diagonal, accents are "one" or "two-piece."
Fractions are Non-locking, with escape on space, period or comma.

Keyboard layout #5 covers the Greek alphabet and a few math signs.

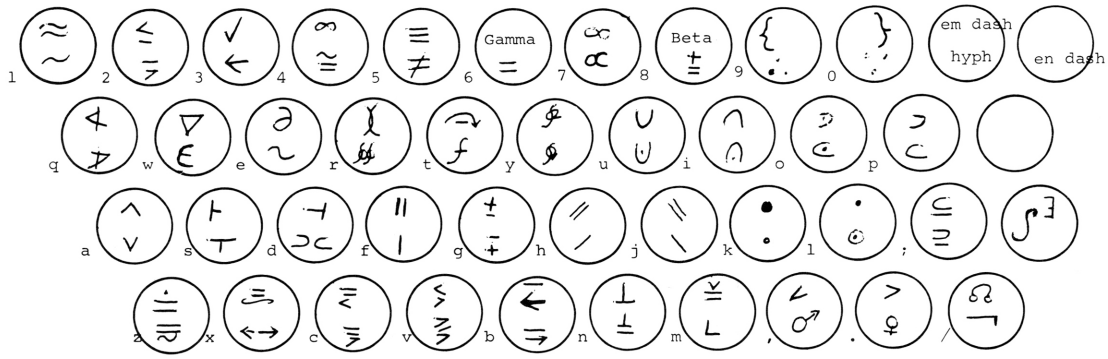



Possible Standard # 5 Layout for Sign Subset
Greek and Some Math

Keyboard layouts #6 and 7 are intended for pi characters and for more technical setting. There is room for other characters. There is also the possibility of adopting other standard layouts which will be more convenient for certain classes of work. (We visualize ten layouts in all).




Possible Standard Layout for Pi Characters #6
Vertical and horizontal rules are of different weights.




Possible Standard #7 Layout for Math and Logic

## Use of Keyboard Layouts for Markup

A font will be designated by a lower-case alpha character. A layout will be designated by a number: such as *1a, or *2a.

For ease in keyboarding, it is possible to move from font a to font b as follows:

```
*laword  *bword
```

The #1 keyboard layout is assumed. But it is not possible to go from #1 to #2 without designating the font, since the next character, probably an alpha character, might be so construed. Hence the convention that on input one either must specify the

font, such as

```
*laword  *6aY  *laword
```

or, to avoid having to think about it, one can use the letter *z* to stand for *the same font you are already in.*

```
*laword *6zY  *lzword
```

Reference may now be had to the sample manuscript copy on the adjoining page, which has been marked to illustrate the manner in which fonts and sign subsets could be designated. The reader should not be intimidated. The illustrations shown are deliberately complex, and some of these conditions would not be encountered frequently.

### The Use of Superiors and Inferiors

The manuscript page just shown ends with a footnote callout reference, which is a superior *1.* It is now necessary to explain how this is coded in our proposed system.

It would be possible to use another letter of the alphabet to indicate a *superior,* and still another letter to label an *inferior.* But we want the ability to distinguish between these conditions and other typographic requirements, and we would also like the ability to position any sign and any graphic in a superior or inferior position, or as a superior to a superior or inferior to an inferior. Therefore we have created a different mode, flagged by a "Beta period" (*.). When this code is encountered the program would look for the next character. If it is an *a* a superior would be called for. A *b* would generate a superior to a superior. A *c* would create an inferior, and a *d* would designate an inferior to an inferior. Parameters would scale the sizes of these characters in relation to the sizes of the standard type faces. For example, if type face *t* were ten point Times Roman, the superior might be designated as six point Times Roman, three points above the base line. The "shift back to base line" command would be "*-" or "Beta hyphen." Whatever sign subset and font you are in, the superior will relate to this. If you wished to change sign subset in the middle of the superior string, this can also be done:

$2^{\text{ième}}$ classe  *la2*.aie*zme*- classe

The accent is self-cancelling. The return from superior puts us back into the preceding condition, which was the *1a font and subset.

### Flags to Designate Format Requirements

We have now endeavored to describe, in a very sophisticated input structure—much more complex and ambitious than any we know of—the way in which changes to fonts and sign subsets may be flagged, *where these shifts may not be inferred from the structure of the data.* The reader may remember that often they may be. Where they may be so inferred the input program would generate the same string of codes we have just used (or the supplier's equivalent version).

*1cCHAPTER I

*dHow to Mark up a Manuscript

*aWe are concerned primarily with the markup codes that relate to changes in type faces and sign subsets. After we have begun the text we need no further guidance until we encounter a character that cannot be found in that subset, or until we want a different typeface or size. Now we are entering italic *bwith these words *aand now we return to roman.

*hThis is a subhead. *aWe have now returned to text roman, after setting the subhead in bold. We now want to typeset the word

e*3zqle*3zeve.

eleve. For further information please refer to 91 5, §3.[1]

*6zy*1z5, *6zY*1z3.*.a

---

In like fashion, format codes may also be inferred, but there are times when they may not be, and we must therefore provide for their inclusion in the input stream or their addition by virtue of correction or update routines.

Some format conditions are one-time situations. Others govern until they are cancelled. An end-of-paragraph code is an example of the former. An indent code is an example of the latter.

What are some of the format conditions we may encounter? Here is a listing of some of the more common ones, and we have proposed certain flags to illustrate how these situations may be described. We shall use the gamma (¢) code to distinguish these conditions from font and subset conditions.

| CONDITION | CODE | POSITION OF CODE |
|---|---|---|
| Determine width of measure | ¢wxxxx | At start |
| End of paragraph (quad left) | ¢l | Following |
| Flush left (quad left) | ¢l | Following |
| Center | ¢c | Following |
| Flush right | ¢r | Following |
| Flush left and right (quad middle) | ¢mm | In between |
| Leader | ¢m.. | In between |
| Indent all paragraphs (fixed value) | ¢ipfxxx | Ahead |
| Indent all paragraphs (variable value) | ¢ipvxx | Ahead |
| Left indent | ¢ilv or f | |
| (fixed value) | ¢ilf | Ahead |
| (variable value) | ¢ilv | Ahead |
| Right indent | ¢irv or f | Ahead |
| Center indent | ¢icv or f | Ahead |
| Hanging indent | ¢ih | Ahead or at definable position |
| Cancel indent (indent ends) | ¢ea, for indent level | After |
| Tab from left to: | ¢t0--9 | Definable position (9 levels) |
| Calculate width of tabular column | ¢xl----¢yl | Before and after |
| Position in tabular column | ¢zl, 2, 3 etc. | Before (9 levels) |
| Go out of tabular | ¢! | Before text, after tabular |
| Hyphenate until further notice | ¢hy | Before |
| Do not hyphenate | ¢nh | Before |
| Justify until further notice | ¢js | Before |
| Set ragged right | ¢rr | Before |
| Set ragged left | ¢rl | Before |
| Set ragged center | ¢rc | Before |
| Do not set (ignore) | ¢ig | Before |
| Resume setting | ¢rs | After ¢ig, before text |
| Add lead above | ¢alxxx | Before |
| Add lead below | ¢lbxxx | Before |

The letter l (ell) is underscored to distinguish from one.

xx or xxx or xxxx indicates values:

- ¢wxxxx - first two x's, picas, last two, points.
- ¢ipfxxx - points
- ¢ilv - first x is ems, second is ens
- ¢alxxx - points
- ¢lbxxx - points

These are not all of the format commands we should need, but they are enough to illustrate the varieties of situations we should encounter. Some of them require explanation. The leader code provides for two characters following the ¢m, and these are the two characters that would alternate to fill the gap. They could both be periods, or a period could be alternated with a thin space, for example.

Indents are visualized as being defined as fixed values, in points, or as ems and ens of whatever point size is called for by the initiating parameters. Hence ¢i/v21 would mean indent two ems and one en. On the other hand, ¢i/f200 would mean indent from the left 200 points.

Indents are also visualized as being "nestable" and hence each indent should be tagged, from *a* to *f*. The code should be ¢i/f200a to designate a first level indent. This can be cancelled by ¢ea for "end indent a." It is also possible to specify the depth that should be indented, at which point the indent becomes self-cancelling. This depth could be expressed either as points of depth or as a number of lines. The code would be ¢i/f20010a for a ten line self-cancelling indent, or ¢i/200x200a for a two hundred point self-cancelling indent.

It is possible to include codes to instruct the program to ignore what follows, and within this area, designated by ¢ig—¢rs, flags for retrieval could be included. Codes can also be used for trial setting, in order to enable the program to calculate widths for tabular setting. For example, if the longest entry in column one is Chillicothe, Ohio, this width would be determined as follows: ¢xlChillicothe, Ohio¢yl.

### Flags to Designate Page Makeup Requirements

For page makeup purposes the following elements must be recognizable:

Chapter opening
Chapter ending
Identification of a Heading (beginning and ending)
Identification of a no-break block (beginning and ending)
Identification of a floatable block (beginning and ending)
Identification of a footnote callout reference
Identification of footnote associated with that reference (beginning and ending)

These are virtually all of the paging conditions needed, except when one wishes to change from a single column to a multi-column format, or vice versa, at a given point in the text.

The following flags could be used:

```
¢#y---------¢#z  around the heading

¢#o for chapter opening
¢#e for chapter ending
¢#n for beginning of no-break area
¢#b for beginning of break area
¢#f for beginning of floatable block
¢#g for end of floatable block
¢#r for footnote reference
¢#s for beginning of footnote
¢#t for end of footnote
```

It would be possible to identify specific footnotes by number or letter, such as ¢#ra and ¢sa------¢ta.

### Use of Global Notations

Generally, strings of codes will be required to define the various characteristics of the text, and these can be described in "macros" or global notations. For example, we will wish to indicate that we are now beginning a chapter (¢#co) with a head (¢#y) which is to be set without hyphenation (¢nh) in a given type face (*1c), and it is to be preceded by ten points of white space (¢la010). All of this can be defined simply as ¢@ha, the macro which will later be defined to include all of these elements. And the chapter head will end with ¢@hb, which will return us to a different type face, put us back in hyphenation, note that the head has ended, and generate white space below.

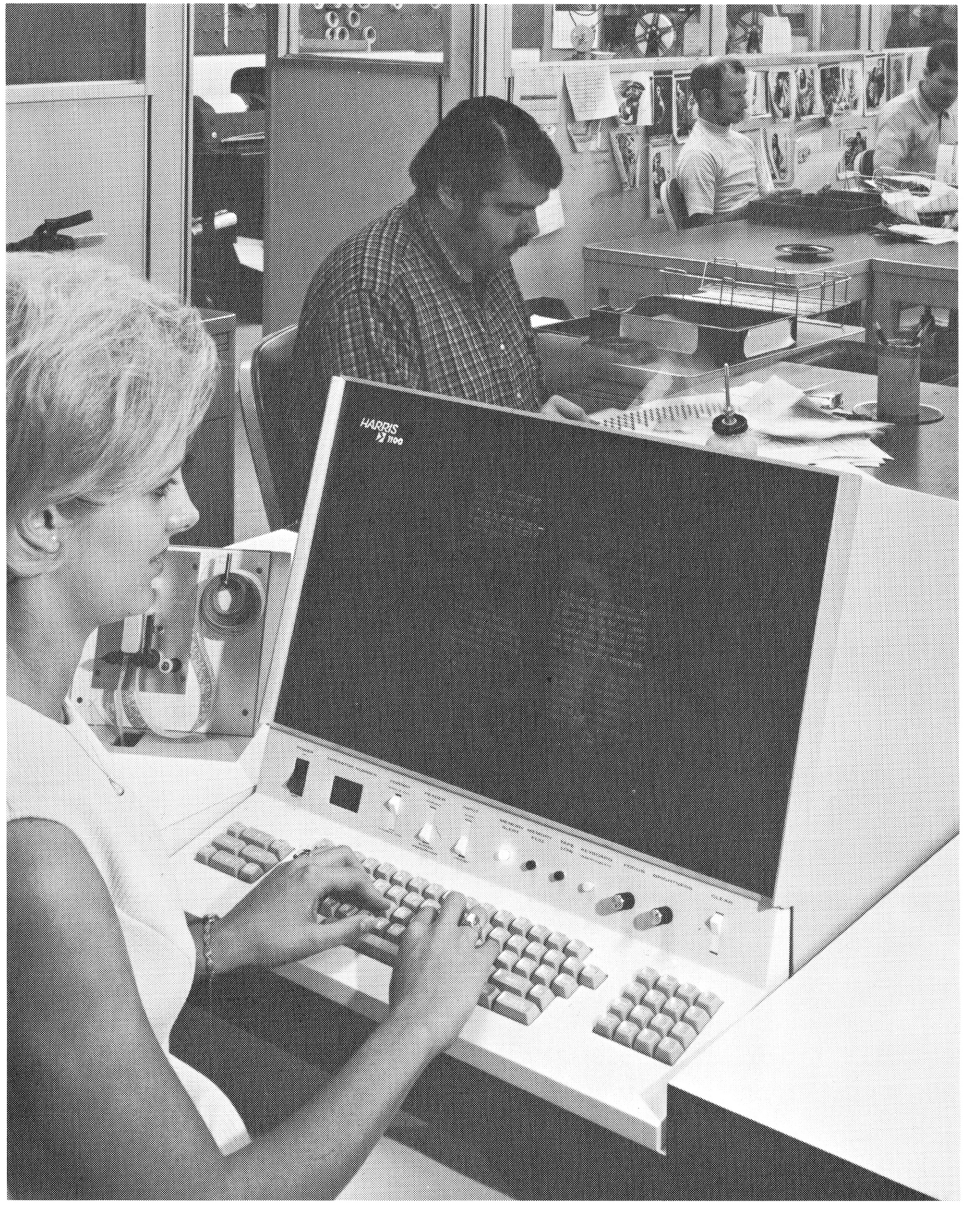
The use of macros or global notations, to designate the conditions encountered in a typical manuscript page, is illustrated on the marked-up manuscript page adjacent to this section. Not all conditions are described, of course, but a number of them are encountered in this example.

### Who Inserts the Coding?

This writer believes that certain standard conditions can be recognized, especially for a given class of work, by the author or by the publisher. Some of the more esoteric coding would have to be inserted by the typesetter. Since global notations can be used widely and can be defined at will, the original input could contain such codes wherever a problem exists. Ultimately, conventions will be established which will be well understood, and when proper diagnostic routines are available to catch inconsistencies and inadvertencies the frightening problem of accurate coding of input may ultimately be overcome. At this time it is not an easy challenge, except for classes of work that conform to a pattern. For large jobs, once the problems are recognized and analyzed, markup becomes relatively easy, and it may be possible to leave most of the decisions to the input keyboarder.

### Review Questions

1. What are the various situations which require coding? Can you remember the writer's classification of these?
2. Why does the author orient his character subsets (sign subsets) around a typewriter keyboard?
3. How would you distinguish between a shift from one font to another and a shift from one character set (sign subset) to another?
4. How do you think superior and inferior characters should be represented?
5. Can you think of any typographic format conditions that have not been described in this chapter?
6. Why is it important to be able to identify these conditions?
7. Under what circumstances is it possible not to mark up the manuscript copy for encoding?
8. What does the author mean when he says that sometimes code strings can be inferred?
9. Why are global statements easier to use than the specific function codes described in this chapter?




ELECTRONIC EDITING AND PROOF READING DISPLAY TERMINAL MANUFACTURED BY HARRIS-INTERTYPE CORPORATION, NOW IN USE IN THE CITY ROOM OF TODAY, THE GANNETT COMPANY'S DAILY NEWSPAPER SERVING THE CAPE KENNEDY AREA.

ART WORK CAN ALSO BE MERGED INTO THE COMPUTER FILE IF IT IS FIRST DIGITIZED. THIS IS THE HARRIS GRAPHIC DIGITIZER WHICH CAN PROVIDE THE CAPABILITY OF COMBINING LINE DRAWINGS, AND ULTIMATELY HALF TONES, WITH THE TEXT STREAM TO BE SET ON THE FOTOTRONIC CRT.

CHAPTER XIX

# The Computer as an Editor

In this chapter we shall be considering ways in which computers can be used to perform certain editorial functions in the publishing industry. It is the author's general thesis that there are economies to be derived from such applications, as well as many other advantages, not the least of which is a substantial saving of time.

We shall be frank to admit that this chapter is the most speculative in this book, because we shall be talking primarily about what *might be* rather than what *is*. Publishers have been singularly unimaginative, and even incompetent, in their explorations of computer applications to the editorial process. On the other hand, it may be that this writer is naive and unrealistic. The reader will have to judge for himself.

## Publishing is a Many-Faceted Thing

At the outset we must recognize that there are many kinds and varieties of publishing activity, even where the end product is a printed book, newspaper or magazine. And nowadays there are many other kinds of "publishing," such as on-line information dissemination, distribution of microforms, audio tapes, video tapes, movies, stills and other photographs, programmed teaching devices, kits, and many other kinds of products and services. If you want white mice, or the skeleton of a beaver, or frogs' eggs to hatch into polywogs, there are "publishers" ready and willing to oblige.

Publishing, we are constantly reminded, is only one aspect of the communications industry, or the knowledge industry, and in this broad context education, amusement, radio, television, libraries, movies, sports, telephones, computers, merchandising, accounting, public relations, government, and almost every conceivable endeavor of mankind is at least tangentially involved. In our book entitled "The Market for Computerized Composition," (Printing Industries of America. Computer Section, 1969) we made the following general comments:

> There is every indication that the printing industry no longer stands by itself, if it ever did, and that it is increasingly a part of the "knowledge," "information," or "communications" industry. Those who wish to be players in this new drama of technological change will have to learn new parts; new techniques and even new art forms are also involved. Whether the successful actors will be the heroes of the past or whether, as in the case of talking pictures, new stars will emerge, while old ones are compelled to seek early retirement, remains to be seen.[1]

We also stated:

> The technological impact of composing-room mechanization during the period 1890–1905 was not without its repercussions—in terms of financial solvency, growth,

[1] As cited, p. 7.

labor relations, human relations and industry-wide adaptation—but the effects of today's revolution may prove much more complex and upsetting.

This will be true in part because the present revolution is more likely to result in a *restructuring of the printing industry* since the origins of the change lie outside of the industry and the industry is obliged to borrow and to make extensive use of techniques, equipment, ideas, inventions and especially personnel from other segments of the economy. (If it does not do so, and in any event, these other elements will invade it anyway.) It will also be true because the impact of these changes will be felt equally profoundly among the customers and users of composition and there will be a fruitful but chaotic interplay and dialogue involving not only old friends wrestling with new problems, but new faces as well.[2]

Publishers, like printers or compositors, are apprehensive concerning their future role. They are curious, but insecure, and are awaiting guidance and instruction. In many cases publishers have been swallowed by electronics firms, or others, seeking to establish a posture in the new information world. Some of these mergers and acquisitions have been cited in our book[3] and we commented there:

In all mergers described above more intensive and more extensive uses of computers are implied. What is probable is that certain materials previously destined exclusively for the book or magazine markets will be processed in a wider variety of ways: for storage and retrieval, for teaching machines, for printing, for "happenings" and for man-computer dialogue.

The kind of dialogue we are referring to is not mere selective retrieval. The computer can be programmed to relate simultaneously to a wide variety of events in such a way that the process of reacting to man's comments and questions may take on a creative aspect.

The ability of the computer to provide individualized responses is stressed as one of its most valuable assets in the teaching process.

We look upon the activities of certain of these large corporations, of government and of not-for-profit institutions, not so much as competition in the sense that they will provide computerized composition as that they will furnish alternatives that may, to some degree, substitute for books and magazines. To the extent that information can be retrieved on demand it is not necessary to publish what people will not need until they want it, and then it is necessary only to give them specifically what they need. And it will be possible to provide certain kinds of information without recourse to typesetting at all. "No stretching of the demonstrated technology is required," says John McCarthy, "to envision computer consoles installed in every home and connected to public utility computers through the telephone system. . . . Everyone will have better access to the Library of Congress than the librarian himself now has. Any page will be immediately accessible . . . Full reports on current events will be available for the asking . . ."[4]

It is logical to suppose that firms having the capability for the storage and manipulation of information in relation to a variety of media and applications will be most advantageously situated with respect to any one of them.

### Editing in Relation to Publishing

Just as there are arguments for the use of computers if you are going to typeset anyway, since there will be by-products, so there are arguments for editing by computer if you are going to use the computer for typesetting. More important, there are arguments for *publishing* via computer if you are going to edit in this fashion or vice

[2] Ibid.

[3] Ibid, p. 106ff.

[4] John McCarthy is quoted from his article on "Information" in the September 1966 issue of *Scientific American.*

versa. Editing is of course only one aspect of publishing, and even editing is a very broad and much misunderstood term.

Editors decide what to publish and what not to publish. They even help to create the supply of manuscripts, by persuading certain authors to make contributions in certain fields. But the kind of editing we shall be discussing in this chapter has to do with:

a.) copy editing, for accuracy, for clarity and for stylistic consistency
b.) verification, to make certain the edited manuscript is faithfully reproduced
c.) typographic interpretation, to make certain the result is represented in a manner which enhances meaning, rather than detracts from it
d.) up-dating, to incorporate last-minute changes
e.) format editing, to assure that the material "fits" the allotted space, or that the space is adequate to accommodate the material
f.) the performance of auxiliary services to enhance the usefulness of the product, such as indexing.

These are only a few of the many functions performed by editors, and in fact some of them are not performed by "editors" at all. They may be performed by "production" personnel, whose responsibility it is to "get the book out," and by proofreaders. Often the duties are sub-contracted to free-lance specialists. We shall not be discussing these six points in logical sequence. However, we shall touch upon each of them during the course of our exposition.

### How Editing Differs From Publisher to Publisher

Even within the narrow areas we have just listed, the function of editing differs as between publishers. Among some publishers, the manuscript is created "in-house" by the staff. Here the publishers are, in a sense, also the authors. This is true of newspapers and many magazines, especially the news magazines. It may also be true of reference works, such as dictionaries and some encyclopedias. It is generally true of directories, product guides, tax services and many trade publications. It may be true of text books and anthologies which become proprietary to certain publication houses.

Where the material is created in-house it can be captured in-house. Where the material becomes part of a data bank, which will be built upon and modified, computer techniques are especially appropriate.

And so we can visualize a number of specific situations in which the editors can relate to the composition process in different ways. Classified ads can be input as they are phoned in; the editorial and business functions of determining where and when to run them can be derived from the initial input. News stories can be captured directly from the wires of the Associated Press, or perhaps even from the reporter's typewriter. Editors can hack at them later when they are in machine-readable form. Or, at the other extreme, a manuscript may be scrutinized by the publisher only in the most perfunctory fashion. He merely serves as a middleman between author and typesetter.

Obviously, then, we cannot present a definitive analysis of how the computer can help the editor. But we can try to be suggestive.

### Functions which can be Performed by the Computer

Here are a number of specific things which could be done:

1. Every word in the input can be compared against a dictionary to see if such a

word exists, and whether it is spelled correctly. Words not in the dictionary can be flagged.

2. Words can be sorted and rearranged in order to diagnose the vocabulary of the author, his consistency in word usage, the degree of difficulty of the material (arranging by word size, for example), excessive and unimaginative uses of the same terms.

3. Synonyms and antonyms can be provided automatically to enlarge the author's or editor's choices.

4. Occurrences of words and phrases can be identified for contextual analysis and for establishing consistency of treatment. Probably the single, most difficult editorial chore is to achieve consistency of style (if this is important!) with respect to compound words. In this book, for example, the reader will have noticed that we have used "photo unit" and "photounit" and perhaps even "photo-unit." We have referred to "hard copy" and "hardcopy devices." We may have said "time-sharing," "timesharing," and "time sharing."

5. References can be flagged and quickly located for revision. Charles Ruggles, the actor, just passed away. How many biographies make mention of him as alive, or in the present tense? Howard Hughes resides at the top of one of his luxury resort inns in Las Vegas, or does he? Walter Hickel is no longer Secretary of the Interior, and Joe Namath is a movie actor as well as a football quarterback. Did the author remember that T. S. Eliot (or is it Elliott or Elliot?) as well as Goldsmith (Walter or Oliver?) wrote a poem with the line "When lovely lady stoops to folly." Somebody else did, too!

Was John Stuart Mill alive when Handel wrote the *Messiah?* Or is it *The Messiah?* Speaking of Charles Ruggles, who gave the Gettysburg address in "Ruggles of Red Gap"? Who directed the movie? Had the principal actor yet starred in "Mutiny on the Bounty"? What was the name of Captain Bligh's antagonist, and what island is associated with that story? Who wrote it, anyway?

How useful it would be, then, not only to have the ability to query the present manuscript but to obtain immediate answers from a huge data file as accessible as the telephone! Obviously, it will require many years to accumulate this kind of information, but the encyclopedia is the obvious place to begin. This has evidently not proved to be an easy undertaking. The *Encyclopaedia Britannica,* in setting its new edition in hot metal, after several years of research on computer implications and applications, evidently found that the state of the arts was not sufficiently well advanced, and yet several significant steps have been taken by them. Meanwhile, at least one other encyclopedia publisher has launched a similar project.

Dictionaries have been compiled with the aid of computers, starting at least with the Random House Dictionary (which was not typeset in this fashion) and continuing on to the recent American Heritage publication, which *was* typeset (in more than one edition) by computer-produced output. Concordances have been generated so that it is not only possible to find which of Shakespeare's plays contains "Fear no more the heat o' the sun," but every other reference, in any play, to *fear, heat* and *sun.* "Fear no more," "weep no more," and "say no more" occur $x$ times in Shakespeare, and $y$ times in Bacon. A concordance of the Bible will locate Beersheba and Bathsheba, the Hittites and the Israelites, in the New Testament as well as the Old, the King James' version as well as the Revised Standard.

6. Automatic indexes can be generated. Dolby and Resnikoff[5] have demonstrated that the extraction of all words (of five or more letters, with certain obvious excep-

[5] "Automatic Indexing of Statistical Abstracts," R & D Consulting Company, Los Altos, Cal.

tions, such as *where, because, however,* etc.) provides more than the raw material for the index, actually producing results perhaps not as good as the best manual index, but probably better than many.

7. Material can be abstracted. IBM has developed an abstracting program called *Syntran* which can elicit the highlights from a legal brief or a court decision.

8. Key words can be identified from pre-determined listings of words (thesauri) associated with particular disciplines. Contexts can be shown and relevance can therefore be readily established.

9. Citations can be classified, cross-referenced, and identifications established between various writers who make use of the same sources.

10. The coincidence or conjunction of terms or words can be detected. For example, whenever *frequency* appears within 50 characters of *distribution* a "hit" is scored.

11. Changes can be effected automatically. Wherever a punctuation mark occurs inside, rather than outside, a closing quote, these two can be interchanged automatically, or else the condition can be identified for further scrutiny.

12. Has the author used "effect" and "affect," "effected" and "affected" correctly? Are there other common failings (the spelling of recommend and accommodate, for example) that ought to be checked routinely and corrected without further deliberation? It is possible to devise an automatic editing routine which scans for the more common failings of all writers, just as it could automatically correct where the "i before e, except after c" rule has been violated. Even split infinitives can be joined together, and prepositions can be queried as appropriate to end sentences with.

The fact is that modern, high-speed computers can perform these routine searches in incredibly short periods of time and at very minimal costs. Is it worth five dollars to go through the entire Bible searching for the word *Gomorrah?* One computer installation we are familiar with charges 23 cents per second of cpu time, and in several seconds it would be possible to search through a 300 page novel!

### Finding Your Way Around in the Manuscript

In our chapter on *corrections,* (Chapter XIII) we indicated various ways in which the text can be addressed (as by line number and spaces within the line) in order to effect changes. We also discussed the problem of how to verify the correction, to make certain that it "took" properly. On-line cathode ray editing devices seem by all odds to be the best way to make changes in the text stream, especially where such changes can be handled in random order rather than serially. Editing techniques of this character require virtually no cpu time because most such devices have their own intelligence—that is to say, corrections can be made within the buffer of the external unit. Adding to, modifying or deleting are relatively easy tasks. Moving blocks of text from one location to another is more difficult. It is necessary to get back to the main frame or cpu, and to take advantage of the editing program stored there.

### The Editing Program

In general, as we have already indicated, there are two types of editing programs now in use. One of them is merely a correction program, of the type we have already discussed. But very sophisticated "editing" programs have also been developed. We are aware of two types of such programs, each possessing its own level of sophistication. *The Administrative Terminal System* is a program formerly marketed as an on-line

service by the Service Bureau Corporation, somewhat ambiguously related to IBM. It was then called *Datatext.* This facility was discontinued and the field was abandoned to others, such as Bowne Time Sharing and VIP, to mention only two, who offer the ATS program (with some modifications of their own) as an on-line, terminal-oriented service. This software is available from IBM to 360/40 (or above) users. Initial input is created via 2741 (IBM Selectric) terminals. Text blocks (files) are identified, can be stored and called back, and then modified. When they are called back they are printed, if desired, on the 2741, in whatever format is prescribed, as to line length, page depth, justified or ragged right. Or the text can be printed at the data center on an upper and lower case on-line printer, and copies delivered to the customer.

Text can be modified by addition or deletion. Lines can be moved. Searches can be made for a unique combination of characters, and global substitutions can often be made for such characters. The output, written to magnetic tape, can then be supplied as input into a typesetting software package.

The reader may be interested in visualizing the way he might work with his Selectric terminal under ATS. He dials a number, waits for a tone and then types his identification. His terminal responds that it is ready and he instructs it to get a certain file. To do this he strikes a certain key that commands "attention." Let us suppose this is an underscore:

```
gChapter 10
```

(get Chapter 10). He ends the message by striking his carriage return.

```
1123 LINES
```

He is told the document has been found and consists of 1123 lines.

```
pan
```

He asks that the document be printed out for him with line numbers.

```
SIGNAL WHEN READY
```

He signals.

```
tf                                  1
This is an example of a printout    2
as entered.                         3
This printout prints all attention  4
actions as well as text.            5
```

Of course he can ask for it to be printed with longer lines, and with double spacing, and with lines justified to provide even right-hand margins if he so prefers.

He decides to move a paragraph which now begins on line 25 and runs to line 50. He wants it after line 67.

```
m67;25;50
```

The terminal wants to make sure. It says:

```
AFTER--We must consider--MOVE--It is contemplated
THRU 50
```

The writer checks and agrees that the twenty-fourth line begins with "We must consider" and that the twenty-fifth line starts with "It is contemplated," so he responds with *y* for "yes". The terminal acknowledges that this change has been made.

He now wishes to replace a word on line fourteen with another word.

```
14;sometimes;occasionally
```

The line is printed out with the change made.

```
He goes there only occasionally
```

The author or editor remembers there should be a period after the word.

```
14;occasionally;occasionally.
```

The terminal responds:

```
He goes there only occasionally.
```

It will be noted that the author or editor has to know how to "talk" through the terminal to the program. He is using a "language" which the program understands. He must become adept in the use of the language and must be able to understand precisely what is required of him and what will happen when he issues his various commands. The terminal is patient and it helps him with queries. It is a willing servant but it considers him rather stupid. For that matter, he considers the computer to be equally unimaginative!

### A More Sophisticated Editing Program

Editing programs have been written which are much more complex than ATS. They have often been devised so that programmers may write their programs with the use of remote terminals. It is possible to say things like this:

If remember, then consider

in which case the program would substitute *consider* for *remember, considers* for *remembers,* and *considering* for *remembering.* Or, one could give the instruction:

*If* Johnson *not* Howard *substitute* The President

in which case every instance of "Johnson" not preceded (say) by "Howard" would be replaced by "The President."

All kinds of logical statements can be posed:

*If* this *and also* that, then—
*If* this *or* that, then—
*If* this *but not* that, then—

Or a command could be given to begin on line 43 and search for a particular character string thereafter, such as "comput" and all occurrences, such as compute, computing, computer, computation, etc. would be identified.

The more complex the program the more complicated the language is likely to be to evoke the desired responses. On the other hand, where certain kinds of tasks are frequently requested, "documents" can be stored which consist of little "programs" which the author or editor can call up and then use to issue the needed commands.

### Editing for Format and Appearance

In some applications, format editing is extensive. In newspapers, stories are generally written so that they can be cut to fit. With news magazines (and many other periodicals) articles are written and re-written to fit around illustrations and to terminate without breaking over to another column or page. *Time,* for example, uses the term "greening" to indicate those portions of a story (underscored in green) which may be deleted if necessary in the interest of brevity or convenience.

Even with books, and especially with encyclopedias, some format editing takes place. If one or two words stand by themselves at the end of a paragraph, for instance, qualifying phrases in the sentence, or in a preceding sentence, may be dropped so that the paragraph "widow" can be avoided. In like manner chapter endings that spill onto a new page are eliminated, and books are made to conform to an even number of signatures.

This type of editing activity cannot be carried out at the input stage. It is necessary to see the product after it has been hyphenated and justified, or even made up into pages. For the most part, existing "editing" programs do not have the capability to handle the output of these composition runs. It is necessary to go back to the initial input stream—a procedure which is both costly and singularly inefficient, since one cannot then predict what the effect of considered changes might be.

Programs for format editing need to be such that an immediate response is engendered. Text already arranged into galleys or pages should be reprocessible, and when changes are made a re-run of the relevant portions of the job should take place instantaneously so that the editor can see what the effect of the change would be.

The CRT display device—an editing terminal that is directly on-line to the computer—is the ideal vehicle for this sort of interaction between man and machine. Some hardware improvements are still necessary, since many of these devices do not offer a large enough character repertoire, especially of upper and lower case letters, and none of them, at this point, offer proportionality, let alone the ability to visualize the copy in the desired type faces. (Simulated italics and bold face may be offered).

There are ways of showing how copy fits by the use of symbolic representations on the CRT display, and these may be adequate for many purposes. A full page of a newspaper can be represented by blocks which indicate identifiable elements. These can be scrutinized more carefully as horizontal lines, to see how the paragraphs break, and then the editor can "zoom in" on individual paragraphs to perform his format revision tasks. Experimental models of such devices have been constructed, but none is operational for English language at this point of time. The Federal Systems Division of IBM, at Gaithersburg, Maryland, has built such a system for a Japanese newspaper.

The editors of *Time* magazine work with computer printouts. They can study the way their paragraphs break. They can count lines and can reprocess their stories as many times as they desire (without having to go back to original input) until they obtain the proper fit. At this writing, the final output is sent over the wire to R. R. Donnelley as paper tape for TTS linecasting equipment, so that the pages, which are

actually fully planned before any type is set, nevertheless have to be assembled by hand from slugs of type.

Existing composition programs are not, generally speaking, re-entrant—that is, they run in a batch-processing mode, from beginning to end. Editing will not come into its own, as a computer-aided function, until all composition programs can be invoked at will, to operate on any portion of the story, so that the editor can see it "as if" the proposed change has been made. When this is possible, and when the other editing functions we have described, having to do with word searches and retrieval of text elements, are also possible, then the computer may truly function as an editorial tool.

Until this is possible, publishers, especially of books, will devote their attention to a production system which will be aimed at the elimination of the present interplay between author, editor and typesetter. The author will be discouraged from tampering with his manuscript once he has surrendered it. The copy editor will be told he must get his instructions right the first time and must send it to the typesetter as a complete package. The typesetter will be asked to provide fully made-up pages and to send them on to the printer, even without review by the editor or the author. This, at least, is the way Leonard Shatzkin, then Vice-President of Manufacturing of the McGraw-Hill Book Company, visualized the operation.

> What is required is that the editor and author, jointly, deliver to the manufacturing process, a manuscript which they are satisfied is complete—the words, the punctuation are the way they want it—the front matter, rear matter, notes are all there—the index entries are all identified—in fact all those decisions that they now normally postpone for six weeks or ten weeks or reconsider later are actually made in the manuscript so that there is no need for any manufacturing step to wait for a new communication or a delayed decision.
>
> That is the first habit that must be broken—postponing decisions, coming back to the same job over and over again to bring it forward another small step.
>
> The second habit we must break is leaning upon the proof—galley proof, page proof, blueprint—for assurance that all is well. We must depend instead on the reliability of the composition system and upon thoroughly verifying the *input to* the system rather than the *output from* the system. . . . . .
>
> Concentrating the attention of author and editor on the manuscript alone—in other words, to getting the input as nearly perfect as they can, a concentration made sensible by the new technology—and then producing the book in the fastest, simplest and most automated manner, is essential if we are to escape from the handicraft stage of book production. It does seem odd that it takes longer to produce a book from manuscript than it does to build a two-family house from blueprint.[6]

Obviously, this is a far cry from the present system, and it is so drastic a proposal that it has elicited moans of apprehension from those charged with editing tasks. In a letter to Victor Strauss, published in *Publishers' Weekly* (July 7, 1969, p. 94) one editor writes:

> . . . how does CRT composition by itself make for a higher degree of quality control, unless we see proofs? Does not the input preparation, given the same high editorial standards, remain the bottleneck? We can and must revise our editing practices if our relationship to the postcomposition phase of production is eliminated, but will not the input-preparation time increase markedly, increasing the bottleneck?

Even the author is hostile to the idea, although the product of his creative mind might find its way to the market while its ideas are still extremely timely. Authors

[6] "Some Effects of Computer Composition On the Editorial Process in Book Publishing," *Publishers Weekly,* April 7, 1969, p. 31.

want a chance to second guess themselves. (It is true, of course, that if the production cycle is foreshortened there will be less need for such reconsideration.) But the author's role and his stake in the enterprise should not be overlooked. Already the editor has had at him with his own stylistic predilections. Victor Strauss writes about this experience as follows:

> This is the first time the author sees his edited manuscript. Now he also sees his work for the first time in print. Even to older experienced writers, galley proofs remain a thrill that cannot be felt by those who never shared this experience. But for a young writer, the arrival of the first proofs of his first book are a great event, possibly even greater excitement than the arrival of the first bound copy.
>
> No pleasure is free of pain. If the galleys are sheer enjoyment, a look at the edited manuscript can become desperation. The beautifully typed pages, on which the Missis had lavished such care, come back full of markings and corrections in different hands and different colors—a real letdown.
>
> Emotion will yield to reason. Eventually the young writer realizes that editing has truly improved his work. He appreciates the great effort and regains his peace of mind. Now he can settle down to the tedious job of proofreading.
>
> Unfortunately, a writer can add a lot of expense during proofreading by making unnecessary corrections. McGraw-Hill, more liberal in this respect than some other publishers, allows the author 20 percent of the original composition cost for corrections. And the editing supervisor provides the author with a 4-page leaflet, "Instructions to Authors," in which author's corrections are extensively discussed and cleverly illustrated. He can see that the addition of a single word can cause the resetting of twelve lines of type.[7]

But copy editors and authors should be re-assured. Some day they will be offered the kind of total involvement in the typesetting process that, contrary to the theory of non-intervention (getting it right and then passing it on)—will permit complete and total participation by everybody at all stages of the process. This awaits the development of re-entrant, interactive, terminal-oriented composition and editing programs. Hopefully, they will not be long in coming.

The reader is cautioned, however, that the present trend among computer typesetters and hardware manufacturers is in the opposite direction. Unable to master the intricacies of present composition programs, they are moving toward smaller systems. Perhaps this is understandable, as a reaction from the over-optimism that characterized initial entry into the field. Bolder concepts will ultimately emerge, but only when more self-confidence has been mustered, and this may take some time.

This writer has suggested that the development of proper programs may be too costly and too difficult to be undertaken by private initiative. He proposed a consortium of interests (governmental, educational and private) at a Composition Workshop held in December, 1970 under the auspices of the National Science Foundation and ASIS, the proceedings of which will be forthcoming shortly. This Workshop was called for the purpose of taking stock of the state-of-the-arts. The interest of the participants in proprietary solutions was too strong, however, for this author's views to prevail. The kind of editing solutions needed will therefore be delayed until someone is brave enough and clever enough to rise to the obvious opportunity. When this has been accomplished, one of the two missing elements will have been provided. The other, of course, will be the willingness of the editors to learn how to use the tools that will then be available to them.

[7] Victor Strauss, "Production Editing of College Texts at McGraw-Hill," a study written for *Publishers' Weekly,* preprinted for the participants in the Newcastle Seminar on Automated Publishing Systems, p. 11.

This "readiness to accept" may not be as hard to win as we sometimes are inclined to think. Younger people are coming along who are not so intimidated by computer technology and programming demands. But even among the existing practitioners acceptance can be gained. One interesting demonstration of this fact can be found in the study by Jerry Carlson,[8] who worked with existing editors of the Farm Journal.

> Generally the response to actual use proceeded from apprehension and nervousness to amusement to serious interest in applying the new machine. It took a little coaxing to get some of the editors to actually try it.
>
> One man editor who had been somewhat skeptical before seeing the machine, although "willing to be a guinea pig if you need one," responded this way after two hours of experience:
>
> > *This is great. The words become much more a physical thing, like you can pound them and hammer them into shape. The best thing is that you are working with clean copy all the time, not a batch of pencil marks. It takes some time getting used to. I composed this page of the book right on there . . .*
>
> Another highly qualified woman editor observed that the machine "will be great for showing you the precise line count any time. And I liked the editing capability—especially the ability to correct a single misspelled character in a word . . . The amazing thing is to think that when a story is done, it's done! Nobody has to retype it. A whole book could be ready for the offset camera just hours after it's written."
>
> Other random comments include these:
>
> "I had thought it wouldn't be useful for editing, which is what I do mostly. But it would be easy to get used to. It's fun to use." . . .
>
> "That's really neat, except the display gets fuzzy around the edges and you have to take a break after about an hour. You can't just sit there and stare at it all day."
>
> "Forces me to edit more severely. Improve technical things like grammar, fitting, widows, etc."

Carlson made some statistical analyses of editors' reactions, and also computed a breakdown of various types of editorial situations, reflecting the ways that editors work. Questionnaires were distributed before and after a demonstration of a specific piece of equipment. "Editors ranked three copy writing and editing systems: (A) present method, (B) a central in-house composing service, and (C) personal use of videotypers. . . . Clearly the editors' present system of writing copy, composing type and correcting proofs . . . was least preferred. . . . System B, the central in-house videotyper service, was placed first by ten people and last by only two." System C was also placed first ten times, but last five times.

Nine editors completed actual use of the machine. After such use personal videotypers were even more strongly preferred than before, with no one moving in a negative direction from his former ranking of System C.

> Editors who used the system often began with the notion that they would have to type clean copy on their first draft. The lingering conviction remains among the more skeptical editors that tedious rewriting would be next to impossible on the videotyper, while the more positive subjects say the reverse is true—the more reworking a piece of copy gets, the more valuable a video terminal becomes.
>
> A correlation . . . indicated no clear-cut relationship between the editors' ages and their attitudinal scores.

[8] "Magazine Writing and Editing on Computer Video Terminals," a Technical Report by Jerry A. Carlson, managing editor, *Farm Journal,* unpublished document, 1970.

Carlson himself expresses a strong preference for the videotyper as an author's tool.

> . . . returning later to correct the errors definitely speeds up the writing process and allows an individual to think of the idea just ahead rather than worrying about the mistake he has just made. In fact, since the videotyper automatically shifts to the next line and the writer does not have to keep alert for the carriage return, a writer can close his eyes and visualize whole sentences in his head. In turn, his fingers seem to type whole words rather than characters. With a typewriter, the end of line is always imposing itself every 34 or 40 characters of magazine copy.

This one study, then—the only one that has so far been made in this area that we are aware of—indicates a surprising amount of open-mindedness among editorial people, so that *readiness to accept* may not be as remote as one might suppose.

But it will still be necessary to offer the editors a *system,* and at the moment a general system does not exist. It will perhaps become one of the major accomplishments of the 1970s.

### Review Questions

1. How do you define "publishing"? Do you know publishers who produce non-book products?
2. What different kinds of editorial functions are there? For what kind of editorial functions would a computer be a useful tool?
3. What are the limitations of present editing programs?
4. Why is it not possible at this time to display a complete page on a CRT terminal device just as it will look in type?
5. Do you think the kind of terminal and the kind of program useful for copy editing might differ from that used for format editing? Why or why not?
6. To use computers for editing purposes, mastery of a "language" to converse with the computer might be necessary for some applications. What kinds of applications would be most likely to require the use of such a language?
7. Some CRT terminals have their own "intelligence." Typewriter-like devices generally use the "intelligence" of the cpu. Why is this?
8. How might editorial uses of computers affect the economics of the composition process?

CHAPTER XX

# Economic Aspects of Computer Typesetting Systems

To be viable, a computer typesetting system must offer an economic advantage. At least it should not be an economic liability. It is understandable, of course, that new installations require some time to "shake down" and during this trial period it is certainly too much to expect that they would pay for themselves. But there must be the prospect of economic gain at some reasonable point of time. We have a right to expect that the practitioners of this art will strive for efficiency—that the computer cannot be asked to perform its miracle in such fashion that it overcomes slovenliness, sloth, inadequate training and overall poor management. But given the will to make the system work, and the application of intelligence and skill, will the system, in the last analysis, offer rewards commensurate with the toil and trouble of installing and mastering it?

There are some who contend that this can never be the case—not only in the field of typesetting, but in many other areas where computers might be introduced. Their reasoning is persuasive, if not compelling: they simply feel that computers take too much high-level talent to be worth the candle. They point out that management, being on the whole ignorant of the technology, delegates the running of the computer center to technicians who are utterly devoid of managerial expertise, and so it shortly becomes a disaster area. There is too much wisdom in this observation for it to be taken lightly. Seldom has there been an activity so fraught with frustrations, where the consequences of non-feasance can readily imply calamity for the entire enterprise. Not long ago we read the annual report of a large corporation whose customary dividend was by-passed simply because a gigantic foul-up of their massive computer installation turned their normally profitable operation into a catastrophe. The reader no doubt can cite many illustrations of his own.

If large and normally efficient corporations who have had years of related data-processing experience (with unit record equipment) can encounter monumental problems in the computerization of relatively straight-forward record-keeping systems, how much more likely it is that typesetters or publishers—notoriously behind the times in their managerial competence, where Harvard Business School graduates have not as yet contributed their diagnostic perspicacity—how much more likely it is, we repeat, that typesetters or publishers, tackling an application ever-so-much more complex and baffling, should fail with a resounding crash that would pull the structure down on top of them!

But, it will be contended, we are overstating the hazards. Not every system, not every application, is a big and risky one. There are modest endeavors, too, well within the competence of the gadget-minded printer. This is, of course, true. We must distin-

guish between the major and minor risks. But the reader will also divine that this writer is attracted to the bolder and more far-reaching experiments: there the risks are truly monumental; but it is only there that the prospect for revolutionary change may be discovered. The financial reward may prove to be proportionate to the odds—which is hardly a recommendation.

In the face of our inexperience, and the complexities of the problem, it would seem expedient to begin with a small and inexpensive system. Such a system might prove, in fact, to meet all of one's requirements; if it did not, one might nevertheless have gained much by an understanding of the application, and be better prepared to move on to bigger things. In the subsequent chapters we shall discuss some of the features and some of the possibilities of small and large systems. In this chapter, while we shall be attempting to deal with economic considerations relevant to either or both, it should be evident that the primary problem we shall be wrestling with has to do with the challenge of a large installation.

## The Economics of Keyboarding

Of course it is true that there are economies to be derived from the keyboarding of input into computer typesetting systems. Close scrutiny of the various elements of keyboarding costs is essential since some of the savings can be realized without the use of a computer at all.

*The Speed of the Keyboard Device.* Input keyboards are certainly more responsive to the speed and the dexterity of the keyboarder than is the case with the manual operation of a linecasting machine. Due to the mechanical features of the latter, the operator is constrained to proceed at a relatively slow rhythm to make allowances for the mechanical requirements of the linecaster. The hot metal must cool, the elevator must elevate, and the matrices must trickle their way back to their home positions. The range of productivity on such a device will generally be found to run from 2500 to 6000 ems per hour, or from 5,000 to 12,000 keystrokes. On relatively simple matter, production at a rate of around 6,000 keystrokes per hour, even of fairly wide measure material, is considered to be fairly satisfactory, if it is quite clean.

On the other hand, a typist, using an electric keyboard, will usually produce within a range of from 40 to 100 words per minute, making some allowance for brief periods of relaxation. If we figure an average of six keystrokes per word (including word spaces) 40 words per minute amounts to 14,400 keystrokes per hour, while 100 words a minute would equal 36,000 keystrokes per hour. It would thus seem that the typist ought to be able to produce roughly three times as much output as the linecasting keyboarder. But with linotype, keyboarding is virtually the end of the process, while for the input keystroker it is only the beginning.

We do not propose to get involved in the controversy concerning whether or not the elimination of hard copy and carriage returns will free the typist sufficiently to permit still another break-through in productivity. There are hardcopy input devices which permit the operator to keep on pounding keys while the carriage is in fact returning, due to the presence of a small buffer which stores keystrokes while the carriage is in reverse motion and puts them out with a frenzy as soon as the carriage resumes its forward direction. There are some who contend that the presence of hard copy is a distraction, whereas others maintain that it gives the operator a sense of competence which contributes to her output speed.

If the typist can go three times as fast as the linecasting machine operator, to what extent do end-of-line decisions deter the latter? Obviously, this is a function of the length of the measure, since short lines are more difficult to fit than long ones. We need some empirical studies in this area, but it may be helpful to advance the hypothesis that about half of the difference in speed between the linecasting operator and the typist may be attributed to the mechanical constraints inhibiting the former, and the balance may be said to be due to the necessity of making end of line decisions. If this presumption is correct, we would expect the speed of the TTS keyboarder—setting justified copy—to fall in the neighborhood of 10,000 to 24,000 keystrokes per hour, the minimum just cited being the more likely standard of achievement. But to advance our hypothesis, let us suppose that, given equal wage costs per hour, input costs might be reduced one-third by the use of TTS and two-thirds by the use of computers. (Obviously this calculation ignores the additional costs occasioned by the necessary hardware to implement the system.)

Suppose we assume lower wage costs for the input keyboarder than for the linotype journeyman. Depending upon the differential, we might find as much as an 80% advantage. In other words, if the labor and fringe rate of the journeyman is five dollars per hour, and if he sets 5,000 keystrokes per hour, the input cost of his work would amount to one dollar per thousand keystrokes, whereas the input of the computer typesetting system keyboarder could conceivably appear to be as low as twenty cents per thousand keystrokes.

But one offsetting consideration is that computer input often requires extra keystrokes for instructions and codes. These would seldom amount to more than ten per cent of the total keystrokes of the job, but an allowance must be made for them.

### Mark-up Costs

Mark-up costs are generally much more substantial for computer input than for straight typesetting or TTS input. We are familiar with some cases in which mark-up, or so-called "programming" costs far exceeded the total value of the job. On the other hand, some of the mark-up which takes place at this point of time must be considered as a substitute for the cost involved in the insertions of pi mats, or the manual assembly of such different page components as extract, footnotes, subheads and the like. Generalizations are extremely difficult because of wide variation in job complexity. Even the very complicated job can, however, be learned with a high degree of efficiency. Therefore if the job is big enough, or if this type of work repeats often enough, mark-up costs can be substantially reduced. The identification of some of the elements to be encoded can be left to the discretion of the input operator. Nevertheless, since we are dealing with very broad propositions, it may be helpful, or at least provocative, to assert that 50% or more of the alleged keyboarding economies can be dissipated by the cost of the more stringent mark-up requirements.

### Proofreading

If proofreaders are expected to read computer printouts rather than galleys, in all probability we can expect an immediate doubling of proofreader costs. Codes which are contained in the text may easily be buried in a computer printout, and one unfortunate keystroke may have consequences far more disastrous than the misspelling of an insignificant word. These non-detected input mistakes become crystal clear when the product is seen in type. Entire pages have been set in italic when only one such

word was desired; a chapter may be set in a hanging indent when only one extract was required to be composed in this fashion. An assortment of pi characters, running on for page after page, makes a fascinating example of mod art, but the author was only expecting a section mark. Whereas some of the diagnostics we discussed earlier, in our chapter on corrections, will cut down the consequences of these errors and should lighten the burden on the proofreader, still it may be said that at this point of time proofreading costs are much greater for computer composition than for hot metal setting. Of course, if the job is typeset two or more times there is no reason why the proofreader's costs should be any greater than in any conventional application, since he is looking at the same materials (galleys and pages in type). But other costs—for computer and photounit time and materials—will of course be higher.

### The Cost of Making Corrections

Correction costs are generally pretty low for linecasting setting. What is involved is usually the insertion of a few slugs of type in each galley. Some work on the linotype machine and some time on the bank are necessary. Occasionally, however, a correction may require the resetting of many lines, perhaps down to the end of a paragraph, because justification is affected.

The same thing is true for filmsetting when computer correction techniques are not employed. The stripping in of these corrections is generally much more expensive than the mere replacement of hot metal slugs.

Whether computer correction capabilities will contribute to the effectiveness of a computer typesetting system will depend upon how good the program is and how expensive it is to run. If only a word needs to be reset, and only that word has to be input, rather than an entire line or the balance of the paragraph, and if the program runs quickly and does not necessitate re-running the entire job or reprocessing portions of the job not affected by the correction, and if there is no substantial idle time in the correction department occasioned by the queuing of re-runs waiting to be processed through the computer, then it may be said that the correction capabilities of a computer typesetting system may be able to contribute significantly to its overall cost effectiveness.

On the other hand, most correction programs that we are familiar with do not meet these criteria. Moreover, there are often many errors introduced by the computer system itself which would not have to be dealt with in a hot metal system or in conventional film typesetting. Examples of this are improper hyphenation and bad line spacing. At this point of time computer correction techniques are more of a liability than an asset, but it is encouraging to consider that there is dramatic room for improvement and ultimately this should be one of the strongest rather than the weakest aspects of the typesetting system.

### Page Make-up

Mark-up costs, where computer systems are involved, are particularly high because of page make-up requirements. Where page make-up is simple, computer programs can do a good job, but hand make-up under those circumstances is usually inexpensive. Where computer pagination programs are most likely to be advantageous is in the calculation and assignment of vertical spacing between text elements. This can be done very precisely and it overcomes one of the major practical objections to film composition because the vertical spreading of numerous lines of type and other com-

ponents requires the cutting and re-spacing of small pieces of film or paper. On the other hand, this kind of task can be done quite efficiently in hot metal. The reader may find it interesting to look at the itemized page make-up costs which appear in E. R. Lannon's study, "Electronic Composition Cost Comparison,"[1] especially at the tables on page 14, which show these estimates:

| *Type of page* | *Labor Cost per Page* |
|---|---|
| "The Group" | $ .0528 |
| "Policy" | .1096 |
| "Prices and Production" | .1177 |
| "Pleistocene" | .1502 |
| "American Bar" | .8959 |
| "Book Catalog" | 1.3642 |
| "Hardware Directory" | 1.3642 |
| "Telephone Directory" | 1.8514 |

These are hot metal labor costs, at union rates, in the Washington, D.C. area, according to Lannon. Those who study the sample pages (initially selected by Jonathan Seybold for his "Report on Cathode Ray Tube Character Generation Devices"[2] —they are reproduced in Lannon's study on pp. 5-12) will recognize the startling fact that those pages which can be most easily put together by computer program are the most expensive to do by hand, and vice versa.

### Author's Alterations

If the author will work from a computer printout he should expect his revisions to cost less, with the assistance of the computer, than when effected through a hot metal system; but if he sees galleys in type (produced through the computer and the photounit) there may not be a great deal of difference in cost. On the whole, however, revisions made throughout a document—especially if they do not involve substantial resequencing of material—should be handled quite economically by a good correction routine.

### The Economics of Scale

Let us presume that keyboarding, proofreading, mark-up and correction costs are variable in character, and largely associated with the application of labor. Let us consider as fixed costs the necessary return for the use of the computer and the programs which drive it, as well as the overhead attributable to the photounit. There are, of course, other costs. Materials and chemicals would be variable in character (although the processor would not) and, in the short run, keyboards would represent a fixed investment.

[1] *A Review of the Costs of Electronic Composition,* Prepared at the direction of the Federal Electronic Printing Committee under the supervision of the Joint Committee on Printing—1969, United States Government Printing Office, Washington, D.C., 1970.

[2] Jonathan Seybold, *Report on Cathode Ray Tube Character Generation Devices,* Printing Industries of America, Washington, D.C., 1969.

For one-time typesetting we will probably find that the variable costs associated with computer typesetting systems may be no less than, and may well exceed by a substantial amount, the variable costs associated with non-computerized typesetting. Fixed costs will of course be much greater in the case of computer technology. It therefore seems evident that it would require a very large volume of production to bring fixed costs down to a manageable per-page expenditure, and even when this is achieved the result may not be competitive with hot metal or with non-computerized filmsetting because the variable or direct costs are too substantial. This generalization is somewhat at variance with the economic analysis of E. R. Lannon, whose study for the Joint Committee on Printing was cited above. What Lannon has found out, in effect, is that *even if* input costs are relatively low, overhead costs are such that breakeven, as against hot metal, could only be achieved on the basis of what is clearly an unrealistic volume of work. Lannon assumes, for example, that there will be no difference in proofreading cost between computer composition and conventional hot metal setting (see page 2 of his report). As we have already indicated, we do not think this is the case at the present time. Although he makes allowance for more keystrokes in computer composition, he assumes an error rate which is more or less commensurate with that encountered in conventional setting, and this, too, is unlikely.

"In this analysis," he writes, "the rate of correction was assumed to be 2.5 lines per 100 lines composed. The correction keying rate was established at 9,000 characters per hour. Initial keying rates were established at an average of 11,590 characters per hour. This difference in keying rates is in keeping with observed practice so as to allow more care in making corrections." [3]

Actually Lannon assigns a correction cost for the labor associated with hot metal which, for a page of "The Group," comes to $.0503. It amounts only to $.0337 for keyboarding of computer corrections. In both cases this represents approximately 3% of the initial keyboarding cost.

Lannon also does not take into account the probable higher mark-up associated with computer composition. The picture is therefore much more dismal than Lannon portrays, and yet he discovers that for the one-time setting of books and the like, break-even requires 37,702 pages of material like "The Group" per month. (See p. 25 of his report.) This figure assumes a Videocomp and a Spectra 70/35. It would be 9,063 pages with a Linotron 505, and 39,813 with the Harris CRT used in conjunction with an IBM 360/30.

Lannon's findings were revealed long before the printed study became available, and at that time computer typesetters were learning the bitter truth for themselves. At first they thought there was something uniquely bad about their own operations, and in a sense, then, they drew comfort from his lucid exposition of the inexorable realities they had wishfully overlooked.

Perhaps we should take a couple of paragraphs to review Lannon's methodology and conclusions in more detail. As we indicated, he selected for his study the same sample pages Jonathan Seybold had selected for his evaluation of the capabilities of CRT typesetting devices. As Jonathan has already shown, there are great differences between these machines with respect to their ability to handle the problem of setting various kinds and styles of pages. Jonathan made some theoretical calculations having

[3] Op. Cit., p. 13.

to do with machine throughput speeds (calculations which were subsequently borne out by Lannon's empirical observations). But Jonathan deliberately did not consider what demands these devices made upon the computer programs and upon the utilization of computer time. It was obvious to him that the ratio of computer time to photounit time could vary a great deal (quite apart from the relative speeds of the typesetting devices themselves) because of the kind of assistance they expected from the computer. He very charitably set these considerations aside and concentrated only upon the output device.

Lannon took the same pages and calculated not only the running time and cost of the CRT unit, but also computer times and costs. To some extent his findings also reflected the efficiency of the software available for use with that computer system. It is conceivable that better, or different, software might lead us to quite different conclusions.

Lannon made a series of calculations in order to reflect several assumptions. In one case he assumes that the photounit is an overhead cost but that computer time can be bought for only those hours needed to produce the break-even volume of work. (Break-even for "The Group" on this assumption is 23,965 pages for the Fototronic, and 24,495 for the Videocomp with a Spectra 70/35.) Alternately he assumes that both the CRT and the computer are fixed investment items which have to be made to support themselves by turning out a sufficient number of pages to bring the cost per page to the hot metal level.

Of course both of these assumptions are unrealistic. If you own or are contractually committed to rent, the only relevant costs are marginal in character. In other words, you are primarily concerned with how much more than your direct costs you will secure by increasing your volume of output. On the other hand, if you do not own or rent the equipment but are considering acquiring it, in a sense these become variable cost items which can be incurred or not. What Lannon was really asking was: can you keep the equipment busy?

The *CIS Newsletter* comments, in this connection:

> This condition—under utilized equipment—appears endemic to the industry. In fact, the Lannon study might be subtitled "The Perils of Over-Capacity." Almost all of the equations break even at very low levels of machine utilization. Typically, the high-speed CRT typesetter is used no more than 20 to 40 hours a month, and the computer less than 100.
>
> Even if we doubled these figures to allow for handling time, job start and stop, extra correction cycles, and general inefficiency, there is still substantial unused time on the machines. Simple economics dictate that users must find some way to turn a profit on these extra hours and thus reduce the overhead burden associated with one-time bookwork, and the per-page cost of doing this work.[4]

[4] Composition Information Services, CIS Newsletter, December 15, 1970, p. 6.

It therefore seems relevant to ask a different type of question: what would my costs be if I only purchased time as I required it, both on the computer and on the photounit? If I have a computer it need not be kept busy solely for typesetting work. If I have a photounit it can also be made available to produce work which is the product of the output of other computers. In this sense then—that is, if we pay only for time used—computer and photounit time are true variable costs. We then can rephrase the question in another way: what proportion of the cost of producing a page consists of labor costs, such as input, proofreading, keyboarding corrections, and the like, and what proportion would be allocated to capital costs, especially the computer and the photounit? This kind of analysis avoids all of the tricky problems having to do with what the economists call the "technical coefficients of production"—that is, the way in which the various elements can be fitted together into a system. (Do I need one computer to keep one photounit busy, or do I need one and one-half?) We shall assume for the moment that we need only an ideal fraction of each. In this connection, people talk knowingly about the ratio of computer time to photounit time when we ought to be talking about dollars. The whole concept of computer "time" is misleading in view of the fact that good programs and good operating systems make it possible to perform a number of tasks simultaneously. How then to allocate costs as between different jobs and uses? We end up by talking only of the allocation of a given size or partition of core of the main frame or cpu for a given number of seconds. We will discover, not surprisingly, that the larger the computer system and the faster the speed of its main frame, the cheaper it is to process a given job, even at higher time rates. If we can buy cpu time and photounit time only as we need it, and still have it available for us with the convenience that we require, there should not be much question about the economic feasibility of the undertaking wherever labor costs and other associated expenses, except for the computer and CRT, are, let us suppose, not in excess of 75% of the equivalent hot metal costs. If we find that we can keep mark-up, keyboarding, proofreading, and other similar activities within reason, then the problem is not whether computer typesetting is economical, but how to discover the peculiar and unique combination of factors which makes it economical.

In a broader sense, there is a more basic question to be asked:

> In a more profound way, the assumptions behind the report, while quite commonly held throughout the industry, may be inimical to the effective use of the new technology. Why should we continue to assume that production procedures and relationships which have been built around the peculiarities of hot metal should continue to remain for computerized photocomposition? [5]

This writer believes that we are driven, inescapably, to the conclusion that a medium-scale computer installation, dedicated to the computer typesetting application, is unlikely to succeed. Others have reached the same finding, apparently, based upon similar analysis and experience; witness the decision of RCA to offer a new solution which utilizes the mini-computer within its Videocomp photounit to perform the logic tasks now assigned to a general-purpose computer.

There appear to be two ways to go: one is to have a small and inexpensive computer used largely or entirely for typesetting purposes. Perhaps this may even be a special-purpose box built specifically to perform composition functions. The other is to buy time on a time-shared "big system."

[5] Ibid, p. 6

Considering the intimate nature of the input problem, and all of the problems of communication which appear to characterize the typesetting function, there is a logical size for the supplier's organization. If he exceeds it he will be inefficient. It would preclude the use of a fast photounit and a medium-scale computer for one's own exclusive needs. There are thus compelling arguments for a slower-speed photounit and a smaller computer.

Of course, even this assumes that the wear and tear on management, the sales force and other administrative practitioners whose services are so essential, is not proportionately greater than with conventional techniques. And there are many who would question this assumption.

On the other hand, the industry can be reorganized so that the supplier no longer assumes input responsibility. The supplier offers only a facility—a capability—a tool. And it would seem logical, with such a concept, where the supplier is no longer concerned with the intimate details of his customer's composition, to provide an on-line, time-shared service over a wide area and to a wide variety of users.

## Special Purpose Applications

We have been considering—whether we knew it or not—the problem of the general-purpose typesetting program as applied to a wide variety of different typesetting jobs. If we move away from this assumption, we can reach entirely different conclusions. There are all kinds of jobs where computer technology has it all over competitive methods. In fact, for some jobs there is no competition at all. They simply could not be produced in type in any other way. Let us consider some of the specialized areas of composition that come to mind:

1. Telephone directories, especially including automatic updating, cumulating, reverse directories (by street address) and re-combinations of elements within the directory, such as by exchange, or businesses, or types of businesses. Especially interesting where the input can be derived from business files, such as service orders.
2. Air-line schedules, where information is computerized in any event, and points of inter-connection must be retrieved because of their effects on the operation of the total system.
3. Library catalogs, especially where the information can be derived from computerized book-ordering systems.
4. Publishers' book catalogs, especially where various combinations are published, and material is required for inventory control.
5. Product guides and industrial directories, especially where information can be elicited from computer-generated questionnaires.
6. Price lists and parts lists, especially where information is maintained for inventory purposes.
7. Abstracts and bibliographic citations, especially where cumulations, sorts and explosions are relevant, and where information can be input during the process of intellectual abstracting.
8. Tariff listings, stock market transactions, bond holdings and other collections of data which require widespread dissemination and constant revision.

It is evident from this suggestive list that there are certain unique features which all of these items have in common:

a.) input costs are not relevant. The information has to be created and maintained at any event.

b.) updating, cumulation, and other data manipulation processes are common.

c.) the information is usually quite massive and repetitive.

d.) typographic or composition demands are often quite simple. Formats, once evolved, do not vary. Special-purpose high-speed composition packages are logical.

### The Editorial Interface

As we have said repeatedly, there is another class of work which seems a logical application, and that is output for the publisher who uses the computer to perform editorial functions. Here, too, input costs are neglible, if they can be said to exist, for the material has to be typed or re-typed many times during the various editorial cycles. Moreover, economies to help support the process will be discovered in the various editing functions which the computer will be able to perform.

### The Large Volume Publisher of a Standard Product

By large volume we mean, of course, a large volume of composition. It should be self-evident that wherever this situation exists—that is, where a lot of composition of a standard variety is anticipated, perhaps accompanied by time constraints, such as with newspaper publishing—that there is an obvious special-purpose application which must be advantageous. If the system has not yet been developed, because of excessive reliance upon generalized and less economical programs, it shortly will be.

### The Creation of a Mass Supply Capability

There may not be an inherent demand, emanating from a single source, but if the supplier is able to develop a specialty for a given class of product he may be able to attract a large volume of similar or identical work. This might even extend to the production of novels or text books, if they are deliberately made to conform to certain standard specifications.

It is logical, therefore, to start with these applications. They may be simple at first, and the supplier who elects to reduce his offerings to a few limited possibilities should not be condemned for so doing. Over a longer period of time, however, we will discover that "standard" applications will encompass greater and greater variety, but these will merely be special cases. Software, which will appear to be uniquely designed, will consist, in reality, of "pluggable" modules which are assembled to meet individual job needs with maximum efficiency.

### Conclusions

Computer typesetting will be advantageous and economical when the logical applications are tackled with the logical equipment and systems. A shaking out process is inevitable. Many people who entered the field did so because they succumbed to the glamor of the equipment and believed, as printers, especially, are too prone to believe, that the secret lies only in equipment. In this application equipment is less important than software, and the most important ingredient of all is the knowledge that exper-

ience and training can bring. Systems must be related to the markets they are intended to serve, but these markets must be discovered and, in some cases, created. Integrating all of the complex aspects of system, equipment, market and people will not be easy. But probably no new field ever is.

### Review Questions

1. Who should be responsible for the management of a computer facility? Is this the kind of function that a highly competent but uninitiated general manager should be asked to tackle?
2. How large a volume of input would be required to justify the installation of a computer renting for $1,000 per month simply on the basis of economies hoped for in keyboarding?
3. Why does the author think that proofreading costs may well be higher in a computer typesetting system? What about mark-up costs?
4. Can you draw any conclusion about the economics of computerized page make-up from the author's discussion of this topic?
5. What is meant by "the economics of scale"?
6. Do you think that the author agrees with Lannon's assumptions about input costs and proofreading costs? What about correction costs?
7. What happens to Lannon's conclusions if variable costs prove to equal the value of the job when produced by hot metal techniques?
8. What assumptions did Lannon make about the utilization of computer and CRT equipment?
9. Do you feel that the author's contention that the treatment of computer and CRT equipment as if these costs were truly variable leads anywhere? If so, where do you think it leads?
10. Why does the author think that the future will lead us either toward bigger or smaller systems?
11. In making an economic analysis of computer typesetting what assumptions should you posit with respect to the organization and structure of the printing and publishing industry—that it will remain as it is, or that it will be drastically changed?
12. What is the relationship between the size of operation which makes sense from a per-unit cost standpoint and the size which makes sense when one considers the intimate nature of most typesetting assignments?
13. What kind of systems might be best suited for special-purpose typesetting jobs?
14. What would you consider a "special-purpose" application as opposed to a "general-purpose" one?

CHAPTER XXI

# An Appraisal of the Extent of Computer Typesetting

Our analysis of the extent to which computer typesetting has developed is based entirely upon statistics gathered by the annual survey of the CIS Newsletter, published most recently as of October 15, 1970. Without the exchange of views and the opportunities to keep abreast of the state of the arts afforded by the publications of Arthur Gardner's organizations we would be ignorant indeed. The entire industry is indebted to him.

Certain cautions are in order in interpreting these data. First, CIS does not have all of the information, although it clearly has most of it. Second, the author may have made mistakes in his compilations, although perhaps not very significant ones in the overall reckoning. Third, it is not always easy to identify a newspaper or a commercial installation judging solely by the name. Fourth, newspapers do commercial work, and vice versa. Fifth, many of these installations use computers not only to drive hot metal linecasters but also to drive photounits, but it is not possible to indicate which output device is the principal one, providing the major justification for the computer. Sixth, we have purposely disregarded the computer-driven Monotype or Monophoto installations. They are not significant in the United States or Canada at this point of time, although there are many more of them abroad. Our discussion of "hot metal typesetting" so far as the United States is concerned, is limited to linecasting; and our listing of second generation filmsetters excludes the Monophoto. Finally, we are including some statistics from firms whose equipment is merely "on order" at this point.

Table I indicates that there are 992 computer typesetting installations in the United States and Canada, more or less. Of these, 708 are in the newspaper industry, and 284, or somewhat more than 28%, are largely commercial, but include also a few captive, institutional or governmental activities. Only 53 out of the 708 newspaper users have exclusively photographic typesetting computer installations, but there are 462 which are exclusively hot metal shops, so far as computers are concerned.

On the other hand, out of 284 non-newspaper applications, only 49 were exclusively hot metal.

| | *Newspaper Computer Typesetting Installations* | | *Other Computer Typesetting Installations* | |
|---|---|---|---|---|
| | No. | % | No. | % |
| *Exclusively hot metal* | 462 | 65.3 | 49 | 17.3 |
| Both metal and film | 193 | 27.3 | 46 | 16.2 |
| Film only | 53 | 7.4 | 189 | 66.5 |
| | 708 | 100.0 | 284 | 100.0 |

TABLE I

COMPUTER TYPESETTING INSTALLATIONS IN THE UNITED STATES AND CANADA
(According to CIS Newsletter)

| Equipment | Newspaper | | | | | Other | | | | | Both | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | |
| Burroughs | 1 | - | - | - | - | - | 1 | 2 | - | - | 1 | 1 | 2 | - | - | 4 |
| DTP, JusTape, Jr., & Linasec | 328 | 42 | 5 | - | - | 32 | 6 | 5 | - | - | 360 | 48 | 10 | - | - | 418 |
| Control Data | 3 | 1 | - | - | - | - | - | 2 | - | - | 3 | 1 | 2 | - | - | 6 |
| PDP-8 | 50 | 56 | 12 | - | - | 3 | 12 | 46 | 1 | - | 53 | 68 | 58 | 1 | - | 180 |
| PDP-10 | - | - | - | - | - | 1 | 1 | 1 | - | - | 1 | 1 | 1 | - | - | 3 |
| Fairchild Comp/ | 6 | 1 | 4 | - | - | 3 | 3 | - | 1 | - | 9 | 4 | 4 | 1 | - | 18 |
| Genl Automation | - | - | 3 | 2 | - | - | - | - | - | - | - | - | 3 | 2 | - | 5 |
| Genl Electric | - | - | - | - | - | - | - | 3 | - | - | - | - | 3 | - | - | 3 |
| Honeywell | 2 | 5 | - | - | - | 4 | 3 | 11 | - | Zip | 6 | 8 | 11 | - | Zip | 26 |
| IBM709/7000 | - | - | - | - | - | - | - | 3 | - | - | - | - | 3 | - | - | 3 |
| IBM 1620 | - | 1 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | 1 |
| IBM 1130 | 47 | 73 | 19 | 3 | - | 4 | 14 | 25 | 5 | - | 51 | 87 | 44 | 8 | - | 190 |
| IBM 1800 | - | 2 | - | - | - | - | - | - | - | - | - | 2 | - | - | - | 2 |
| IBM 360 | 12 | 10 | 1 | - | 1 | - | 3 | 17 | 1 | 26 | 12 | 13 | 18 | 1 | 27 | 71 |
| Intertype | 9 | - | 3 | - | - | 1 | - | 2 | - | - | 10 | - | 5 | - | - | 15 |
| NCR | - | 1 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | 1 |
| RCA 301 | 3 | 1 | - | - | - | - | - | - | - | 1 | 3 | 1 | - | - | 1 | 5 |
| RCA Spectra 70 | 1 | - | - | - | - | 1 | - | 4 | 1 | 14 | 2 | - | 4 | 1 | 14 | 21 |
| Univac | - | - | - | - | - | - | 3 | 12 | 4 | 1 | - | 3 | 12 | 4 | 1 | 20 |
| Total | 462 | 193 | 47 | 5 | 1 | 49 | 46 | 133 | 13 | 42+ Zip | 511 | 239 | 180 | 18 | 43 +Zip | 992 |

Column Identification: Lino=Linecasting; "2 G Ph"=second generation photosetter; "505" is Linotron 505; "HS" is High-Speed CRT photocomp. The one Photon "ZIP" is separately identified.
Summary does not include two additional Photon Zips, one run from IBM 360/40 and one from a Spectra 70.

TABLE II

COMPUTER TYPESETTING INSTALLATIONS IN THE UNITED STATES AND CANADA
(According to CIS Newsletter)
EXCLUDING SPECIAL-PURPOSE INSTALLATIONS

| Equipment | Newspaper | | | | | Other | | | | | Total | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | |
| Burroughs | 1 | - | - | - | - | - | 1 | 2 | - | - | 1 | 1 | 2 | - | - | 4 |
| Control Data | 3 | 1 | - | - | - | - | - | 2 | - | - | 3 | 1 | 2 | - | - | 6 |
| PDP-8 | 50 | 56 | 12 | - | - | 3 | 12 | 46 | 1 | - | 53 | 68 | 58 | 1 | - | 180 |
| PDP-10 | - | - | - | - | - | 1 | 1 | 1 | - | - | 1 | 1 | 1 | - | - | 3 |
| Fairchild | 6 | 1 | 4 | - | - | 3 | 3 | - | 1 | - | 9 | 4 | 4 | 1 | - | 18 |
| Genl Autom. | - | - | 3 | 2 | - | - | - | - | - | - | - | - | 3 | 2 | - | 5 |
| GE | - | - | - | - | - | - | - | 3 | - | - | - | - | 3 | - | - | 3 |
| Honeywell | 2 | 5 | - | - | - | 4 | 3 | 11 | - | Zip | 6 | 8 | 11 | - | Zip | 26 |
| IBM 709/7000 | - | - | - | - | - | - | - | 3 | - | - | - | - | 3 | - | - | 3 |
| IBM 1620 | - | 1 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | 1 |
| IBM 1130 | 47 | 73 | 19 | 3 | - | 4 | 14 | 25 | 5 | - | 51 | 87 | 44 | 8 | - | 190 |
| IBM 360 | 12 | 10 | 1 | - | 1 | - | 3 | 17 | 1 | 26+ Zip | 12 | 13 | 18 | 1 | 27+ Zip | 72 |
| Intertype | 9 | - | 3 | - | - | 1 | - | 2 | - | - | 10 | - | 5 | - | - | 15 |
| NCR | - | 1 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | 1 |
| RCA 301 | 3 | 1 | - | - | - | - | - | - | - | 1 | 3 | 1 | - | - | 1 | 5 |
| RCA Spectra | 1 | - | - | - | - | 1 | - | 4 | 1 | 14+ Zip | 2 | - | 4 | 1 | 14+ Zip | 22 |
| Univac | - | - | - | - | - | - | 3 | 12 | 4 | 1 | - | 3 | 12 | 4 | 1 | 20 |
| Total | 134 | 149 | 42 | 5 | 1 | 17 | 40 | 128 | 13 | 42+ 3 Zips | 151 | 189 | 170 | 18 | 43+ 3 Zips | 574 |

The figures in Table I, from which the above statistics were compiled, are misleading in one sense. They include special-purpose computers as well as general-purpose computers used for typesetting purposes. And some of these special-purpose computers perform tasks which do not differ substantially from the "hyphenless justification" performed directly by certain photounits, or from the counting properties of certain keyboards. And one of the computers was designed only to perform certain functions in the inputting of data for telephone directories. Accordingly, to get a more realistic picture concerning the application of "computers" to typesetting, we have decided to eliminate the Linasec, JusTape, JusTape Jr. and Compugraphic DTP statistics. It should be understood, however, that no reflection upon this equipment is intended. The usefulness of the equipment is certainly evident from the very large number of installations. Table II, therefore, presents the same data as Table I without the inclusion of these special devices. We now find that, so defined, there are 564 computer typesetting installations in the United States and Canada, of which 60% are in the newspapers.

| | *Newspaper Computer Typesetting Installations* | | *Other Computer Typesetting Installations* | |
|---|---|---|---|---|
| | No. | % | No. | % |
| Exclusively hot metal | 134 | 40.5 | 17 | 7.0 |
| Both metal and film | 149 | 45.0 | 40 | 16.5 |
| Film only | 48 | 14.5 | 186 | 76.5 |
| | 331 | 100.0 | 243 | 100.0 |

Thus we may observe that in the newspaper industry computers are used either for hot metal setting or for both hot metal and film, whereas there is a neglible amount of "hot metal only" setting in the commercial industry, while the largest proportion is exclusively photosetting.

Reference to the two tables, especially Table II, will also show that most of the high-speed photo devices are found outside of the newspaper industry. There is only one high-speed CRT in the "newspaper" industry, and this machine was not installed for news setting. There are five Linotron 505's. On the other hand, there are thirteen 505's, 42 high-speed CRTs, and three Zips outside of the newspaper industry. Of course some of these are government installations, but most of them are not.

The two most popular computers are clearly the IBM 1130 and the PDP-8. Out of 190 installations of 1130's, there are 142 in newspaper establishments, or about 75%. Of the 180 PDP 8's, 118, or about 65% are found in newspapers. Sixty out of 72 IBM 360's are in the commercial industry—at least, outside of newspapers. And twenty out of 22 Spectra 70's are also outside of newspapers.

We may see, therefore, that newspapers tend to use smaller computers, cheaper photounits, and programs that will also embrace linecasting. Commercial shops lean more toward larger computers, high-speed photosetting devices, and exclusively film or paper composition.

This is not too surprising, even if we assume newspapers may have a greater ability-to-pay and a larger built-in demand for composition. The larger papers are still wedded to metal setting because of printing and platemaking technology. Newspaper typographic demands are also not so varied as those in the commercial industry, and

where demands require great flexibility in character repertoire, type sizes, styles, etc., photosetting is more logical than metal. But software requirements are, of course, that much more sophisticated.

We can also infer, from these tables, that newspapers have moved toward photocomposition from hot metal, but most commercial installations never even tried computerized hot metal setting. Newspapers tend to use computers and computer programs that were originally devised for linecasting, and were then adapted for film setting. Commercial shops tend to use computers and programs that were especially conceived, in this industry, for photo composition.

If we make the arbitrary assumption that every photocomposition installation shown represents only one typesetting device, and if we further assume that the ratio of effectiveness may be represented by 20 lines per minute on second generation devices, 100 lines per minute on the Linotron 505, 300 lines per minute on the Photon Zip and 1,000 lines per minute on the high-speed CRT devices, then we get the following picture of capacity per minute:

| *Newspapers* | | *Other* | |
|---|---|---|---|
| *Device* | *Lines* | *Device* | *Lines* |
| 191 second generation photo units x 20 | 3,820 | 168 second generation photo units x 20 | 3,360 |
| 5 Linotron 505's x 100 | 500 | 13 505's x 100 | 1,300 |
| 1 high-speed CRT | 1,000 | 3 Zips x 300 | 900 |
| | | 42 high-speed CRT's | 42,000 |
| | 5,320 | | 47,560 |

On the basis of this kind of comparison the reader will be able to understand why this writer has devoted so much time and attention to the larger computers and the faster devices. While we recognize that there is presently a trend toward smaller computers and slower devices, the larger systems have so much more capability, at this point of time, that they will simply have to be reckoned with in the market place, unless these firms are unable to stay in business.

## Installations Abroad

Table III also excludes the JusTape type of installation (as does Table II), so that 54 such installations, of which 37 are hot metal newspaper installations and five are hot metal commercial, are not included. The following summary indicates the salient facts:

| | *Newspapers* | | *Other* | |
|---|---|---|---|---|
| | No. | % | No. | % |
| Hot Metal Only | 128 | 65.3 | 20 | 20.6 |
| Both Metal and Film | 35 | 17.9 | 10 | 10.3 |
| Film only | 33 | 16.8 | 67 | 69.1 |
| | 196 | 100.0 | 97 | 100.0 |

Out of the total number of photosetting installations, the 196 in the newspaper industry represents 67% of the total.

TABLE III

COMPUTER TYPESETTING INSTALLATIONS OUTSIDE OF NORTH AMERICA

| Equipment | Newspaper | | | | | Other | | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | Lino | Lino + 2 G Ph | 2 G Ph | 505 | HS | |
| Bull-GE | 2 | - | - | - | - | - | - | 3 | - | - | 2 | - | 3 | - | - | 5 |
| Control D. | 1 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | 1 |
| PDP-8 | 43 | 12 | 8 | 1 | - | 5 | 1 | 20 | 3 | - | 48 | 13 | 28 | 4 | - | 93 |
| IBM 1130 | 30 | 14 | 6 | 5 | - | 3 | 3 | 5 | 1 | - | 33 | 17 | 11 | 6 | - | 67 |
| IBM 360 | 3 | 7 | 1 | 1 | - | 5 | 3 | 2 | 1 | 1 | 8 | 10 | 3 | 2 | 1 | 24 |
| ICL | 3 | 1 | 2 | 4 | - | 2 | - | 5 | 2 | 2 | 5 | 1 | 7 | 6 | 2 | 21 |
| Intertype | 3 | - | - | - | - | - | 1 | 1 | - | - | 3 | 1 | 1 | - | - | 5 |
| Marconi-Ell. | - | - | - | - | - | 1 | 1 | 5 | 1 | - | 1 | 1 | 5 | 1 | - | 8 |
| Muirhead | 22 | - | - | - | - | - | - | - | - | - | 22 | - | - | - | - | 22 |
| NCR | - | - | - | - | - | | - | - | - | 1 | - | - | - | - | 1 | 1 |
| Philips | 11 | 1 | - | 1 | - | 3 | - | 5 | - | 1 | 14 | 1 | 5 | 1 | 1 | 22 |
| Siemens-H. | 10 | - | - | - | 2 | 1 | 1 | 2 | - | 5 | 11 | 1 | 2 | - | 7 | 21 |
| Univac | - | - | - | - | Zip | - | - | - | - | 1 | - | - | - | - | 1+ Zip | 33 |
| Total | 128 | 35 | 18 | 12 | 2+ Zip | 20 | 10 | 48 | 8 | 11 | 148 | 45 | 66 | 20 | 13+ Zip | 293 |

Note: Linasec, JusTape and JusTape Jr. not included. Nor are approximately 8 Guttinger GSA Monophoto installations.

In terms of capacity, the following table will be helpful:

| *Newspapers* | | *Other* | |
|---|---|---|---|
| *Device* | *Lines* | *Device* | *Lines* |
| 53 second generation photounits x 20 | 1,060 | 58 second generation photounits x 20 | 1,160 |
| 12 505s x 100 | 1,200 | 8 505's x 100 | 800 |
| 1 Zip x 300 | 300 | | |
| 2 HS CRT's | 2,000 | 11 HS CRT's | 11,000 |
| | 4,560 | | 22,960 |

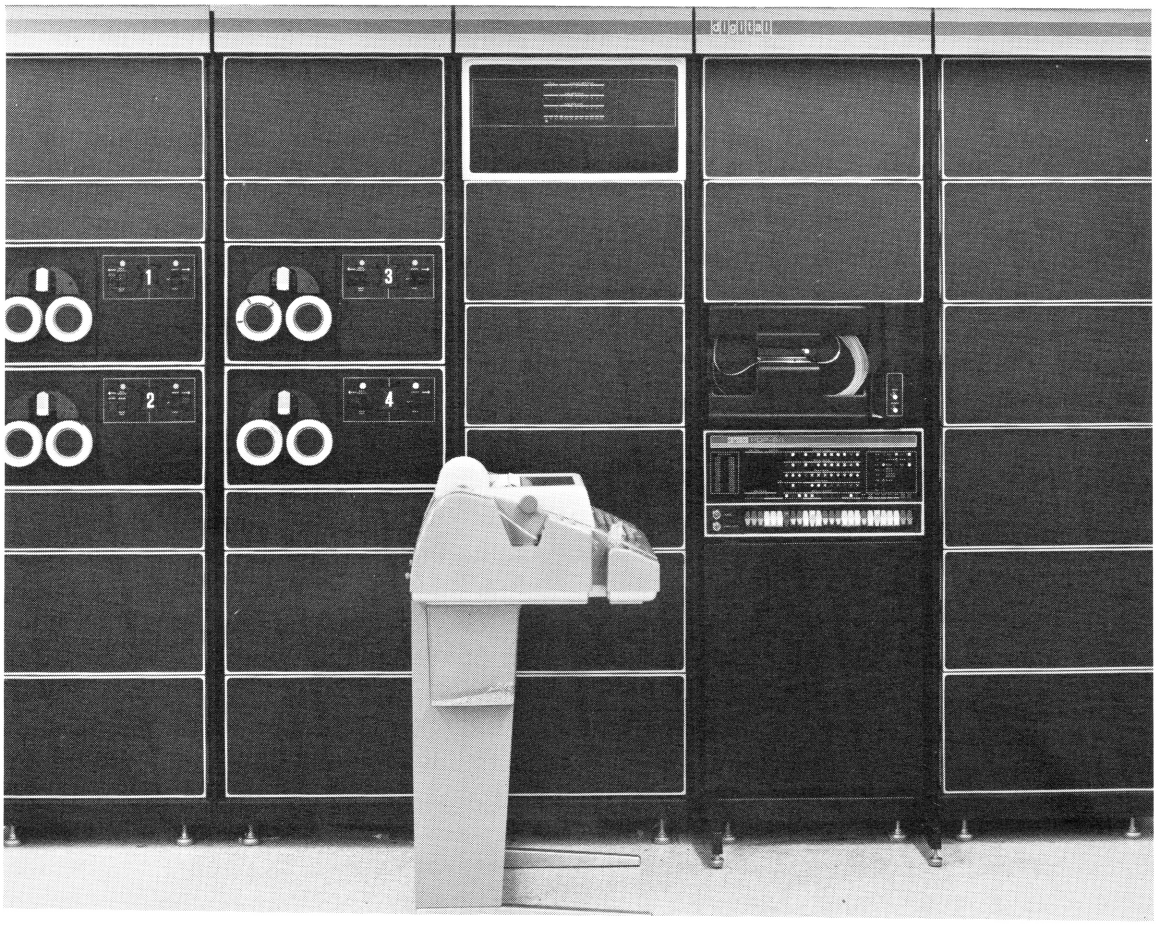
Thus, outside of North America the same conditions seem to apply: high speed devices are more frequently found in non-newspaper installations. The device most frequently found is the Siemens-Hell Digiset. The PDP-8 and the IBM 1130 are popular computers, but larger devices are used to drive the high speed machines.

There are many second-generation photounits on the market nowadays, and their prices continue to fall, while their speed and capabilities increase. For the most part, however, these are not computer-driven. One thing is very clear: the proportion of photocomposition to all forms of composition is increasing substantially, even to the point where the manufacture of hot metal devices is being phased out altogether. Whether the proportion of computer-driven photounits to all photounits is increasing may be a matter for speculation, but it seems probable that, despite the availability of low-cost non-computerized machines, the capacity of the computer-driven units will surely result in a steadily-moving ratio of composition being transferred to computerized photocomposition.

The trend will also be enhanced by virtue of the fact that manufacturers are now revamping their machines so that the computer which is needed to service the internal requirements of the unit (storing width values, calculating flash times, holding in core the digitized fonts needed) will also perform most of the logical composition functions. Yet some of these machines will also be capable (as the Linotron 505 is now) of being driven directly from pre-justified input tape. This will be especially useful for the making of last-minute corrections. The Lannon study certainly influenced RCA to develop plans to offer software for use on the 1600 controller for the Videocomp. A new approach for the Linotron 505 has been announced in England, along the lines of the experimental model created at the University of Newcastle-upon-Tyne, which substitutes a mini-computer for the controller of the 505, so that it becomes much more versatile. Auto-Logic, a subsidiary of Alphanumeric, has similar conceptions under study, which may or may not be associated with the marketing of the Photon 7000—a lower-cost high-speed CRT device which Photon will push as its first tube typesetter.

It is reasonable to predict, therefore, that computer typesetter overhead will decrease. Machines may become a little slower, but with the less awesome investment which will be required, and the probable availability of supportive timesharing facilities, we can certainly anticipate that computer-related or computer-aided photocomposition will take over a larger and ever-larger share of a growing market.

As we have indicated, there will be new applications and new devices, and they will warrant revisions in this text. But for the time being it seems most appropriate to devote the next two chapters to a discussion of the kinds of installations and devices in most common use today (January, 1971). An inspection of the tables contained in this chapter leave no room for doubt that the IBM 1130 and the PDP-8, with second-generation output units, are by far the most prevalent, and so the next chapter will consider the characteristics of 1130 and PDP-8 software in greater depth. Chapter XXIII will provide more detail concerning the hardware and software characteristics of existing high-speed CRT systems.




THIS IS AN ILLUSTRATION OF A PDP–8, A PRODUCT OF THE DIGITAL EQUIPMENT CORPORATION. TWO DUAL-UNIT DECTAPES AND THE ASR 33 TELETYPE ARE SHOWN.

CHAPTER XXII

# Small-Scale Computer Typesetting Systems

As we have seen, the Digital Equipment PDP-8 and the IBM 1130 are beyond question the most popular computers in the field of typesetting at this point of time (early 1971), if we disregard special-purpose hardware such as the Linasec or JusTape. These general-purpose computers are "small" in the sense that the programs are not as ambitious and far-reaching as those which have been written and made available for such medium-scale (and relatively high-priced) computers as the IBM 360/30 or 40, or the RCA Spectra 70/35 or 45. As a rule, the smaller computer systems do not include (with the supplied software, at least) any sort of general-purpose pagination solutions. Even correction passes are non-existent or may be relatively limited in scope. And yet these smaller machines can perform a wide variety of functions and are especially efficient when coupled with second-generation typesetting devices. It is not so logical to combine them with third-generation CRT machines because the high typesetting speeds of the latter virtually require paginated output.

There are economies to be derived from the use of these small systems. In some cases these economies may be dramatic. Sometimes they are realized with some impairment of the quality of the typeset product, but an impairment which is not resented in the light of the cost advantages which accompany the transition. In other instances they have actually presented the opportunity to realize both an upgrading of the product and a lowering of the cost of producing it. There is, moreover, no need to assume that experience with such systems need have disastrous economic consequences. The risks are relatively small and the gains can be quite substantial.

The widespread use of these smaller systems will ultimately have an impact upon the market for hot metal typesetting. In a speech which the writer delivered at the American University Institute on *Technological Change and Administration in Printing, Publishing, and Information Services,* February 8-10, 1971, he made the following comments:

> I happen to believe that a comparison such as Lannon's, which tests the economic viability of a computer typesetting system against hot metal linecasting, assumes that the pricing of book work, or similar setting, will be determined for a long time to come, by hot metal prices. This, I submit, will not be the case. My expectation is that the pricing for book and periodical composition will be determined by the cost of setting photocomposition on very cheap second-generation equipment, partly or wholly automated with mini-computer special-purpose logic elements, and that these costs and prices will be substantially lower that hot metal costs and prices. This would mean that the high-speed computer systems presently in existence, with their medium-scale computers, would find their break-even point, even under the most optimistic input assumptions, not at something like 40,000 pages a month, but at 60,000 to 80,000 pages, if such a volume could ever be achieved.

Soon, however, it will be difficult to discuss the role of the computer in photocomposition (as distinguished from the advent and impact of photocomposition itself) because the two processes are rapidly becoming inseparable. Phototypesetting devices (even second-generation ones) are now able, in some cases, to perform the logical and arithmetical functions of hyphenation and justification. It is not therefore necessary to settle for "hyphenless justification" if one elects to forego the computer on one hand and the preparation of pre-justified tape (by means of the counting keyboard) on the other. For example, the Compugraphic Model CG4961, which sells for $10,500, accepts unjustified or justified tape, and if the former, the machine itself justifies and hyphenates. Sales Manager Harold Evans informed this writer (February 3, 1971) that of the 3500 or 3600 Compugraphic photounits presently installed or about to be installed in the United States, roughly one-third of the sales are of 4961s, and it is anticipated that many of these will be fed unjustified tape. Mergenthaler offers a VIP typesetting machine in the $20,000 range, with functions in the same general manner, and Harris-Intertype's TxT, at around $50,000, also has the same capabilities. Although no machines have yet been delivered by Harris, some 40 or so are rumored to be on order.

But our attention will be focussed upon the general-purpose computer, operated in conjunction with a photounit, and, as noted, especially upon the PDP-8 and the IBM 1130.

As between these two, it may be said that only the PDP-8 deserves to be called a "mini" computer, while the 1130 is perhaps best described as a "small" computer.

### What is a "Mini"?

Immediately we are involved in matters of definition. Probably the only safe definition of a mini-computer is that it is small and relatively inexpensive. Often its instruction set is as extensive as that of many larger machines. Usually it is as fast (cycle time, or the time required to execute the average instruction), and often it is faster, than many of the larger and more expensive computers. Frequently it is possible to interface with as many peripheral devices, and to add modules of memory, and often these interfaces are "standard" and achieved with no unusual difficulty. Usually, but not always, such computers are sold only as "OEM" items (that is, "original equipment manufacturer" or wholesale items to other manufacturers) but this is not true of all mini manufacturers. For example, Digital sells in both markets.

In general, "minis" are desk-top items, and their core usually runs from 4k to perhaps 16k, in 8, 12 or 16 bit words. The price of the main frame of a PDP-8E/CA, with 4k core and teletype control, is currently quoted at $5,640. This is total purchase price, not monthly rental. One authority seems arbitrarily to define a "mini" as any computer whose main frame customarily sells for under $20,000.[1] Some think of "minis" as special-purpose devices, and it is true that in the OEM market they usually serve this function, replacing hardwired functions with more flexible software. But they are truly general-purpose computers.

Digital was one of the first mini-computer manufacturers to make an impression upon the market, especially with a typesetting application. Many other manufacturers

[1] Karl Hinrichs, "The Impact of Mini-computers on Industry." Computers and Automation, December, 1969, p. 28.

have followed with their own "minis,"—some of them even designed or marketed by persons who were originally associated with DEC. They offer equipment which appears to be roughly comparable (or even more exciting) at prices which are often even somewhat lower.

Typesetting programs and systems for the PDP-8 are available directly from the Digital Equipment Corporation (146 Main Street, Maynard, Mass., 01754). In addition, Digital has made arrangements with certain other companies to sell their products with that other company's proprietary software. For example, a company known as CSI, or Composition Systems, Inc., of White Plains, N.Y., offers various typesetting packages, all of which utilize one or another configuration of the PDP-8, in conjunction with such typesetting devices as linecasting, or Linofilm, or a Photon 713 or 560, or a Linotron 505, to name a few of the current packages. Information Control Systems, located in Ann Arbor, Michigan, sells a PDP-8 input and editing system known as "Astrocomp." Such suppliers derive their income not only from their software, but from the sale of the hardware as well.

This author will not express an opinion as to whether the software supplied directly by Digital is equal to or better than that furnished, say, by CSI. He has heard arguments on both sides of this question and at the moment is in no position to evaluate them. Moreover, a great deal of the value of any system lies in its adaptation to the customer's special requirements. However, it may be helpful to the reader to be aware of certain of the "standard" packages and configurations which Digital itself advertises.

### Digital Typesetting Systems

For example, there is a computer configuration which implements a single-pass typesetting system and which sells for $11,845 (December, 1970 advertised price). It consists of the 4k main frame, a 110 cps paper tape punch, a 140 cps paper tape reader, and an ASR 33 Teletype, which acts as the computer console and communicating device. The software program runs (assuming not too frequent hyphenation) at a speed which tends to keep the paper tape punch busy (hence a throughput speed of 110 characters per second), uses a 250 word exception dictionary, processes "all standard indentions" and can handle "over 20 combinations of column width, font, and type size." Three or four different tables of set width values can be handled, with a "mixing capability for two magazines."

For a little more money ($16,550) the system is modified to handle two paper tape readers and two paper tape punches, with additional "allotting capability," making it possible (for more money, of course) to accommodate up to 16 tape readers and 16 tape punches. These devices would not overlap in their use since the computer is of course not multi-programmed, but would permit various classes of input or output to be accepted and delivered to appropriate stations.

In neither of the two systems just described is it possible to provide interim storage of input. These two configurations imply the operation of "torn tape" systems, which means paper tape in and out in one pass, under operator control. But for a total of $27,650, a "dual unit DECtape" is brought into the system, with its controller and cabinet. Even with the 4k core, the use of the DECtape permits loading of additional tables of width values as needed ("over 100 combinations of column width, font, and type sizes" as well as "30 or more non-unit-count font tables.") In addition, the exception dictionary jumps to 5000 words. Here again, however, there is no interim storage of input for subsequent updating and correction.

DECtape is a non-standard and very special kind of magnetic tape with certain indexable characteristics which makes it both an inexpensive tape transport and permits its use somewhat in a random access capacity, since the entire tape does not have to be searched, character for character or block by block, as is true of conventional tape systems. Hence using DECtape for accessing exception words or width tables does not slow down the system, especially when it is punch-bound at any event.

Another alternate system, costing $34,580, includes a 32k disk file and control, as well as the dual unit DECtape, and it appears to be directed specifically toward the photocomposition market since it is claimed that it "provides high speed processing of both text and display photocomposition programs," which means, presumably, that it can bring more typographic functions into core for manipulation, and it can also handle a greater variety of type fonts. But, as we shall see, even the cheapest and smallest system will work with photocomposition devices for limited varieties of relatively standard work.

For $23,480 you can buy a similar system with the disk file but not the DECtape.

Finally, there are two (more expensive) systems which offer "storage and edit" capability. One costs $64,350, enlarges memory by 4k, adds a 250 line-per-minute printer for proofing purposes, a second dual-unit tape transport, and a 262,144 word disk file. The other costs $72,550, also has the memory extension feature (a third 4k module is recommended for another $3,000), but has only one DECtape dual unit. However, in addition to the line printer and the 262,144 word disk file, it also provides disk storage of 841,488 words, with a controller which will accommodate three such disks. Interim storage can thus be had either on DECtape or on disk, depending upon how much storage is needed and how accessible it has to be.

The "edit" programs vary in their complexity from one which requires the substitution of an entire line to one which permits feeding in only the word to be changed.

The procedure followed by the systems with the edit capability involves the reading in of paper tape, performing hyphenation and justification, writing (for example) to the disk, printing line-for-line in upper and lower case (additional graphics can also be provided as part of the proofing character repertoire), usually showing typesetting function codes, dumping from disk to DECtape if needed, or maintaining a store on disk while outputting a paper tape for photounit proofing and inspection. Corrections can then be accepted from paper tape (the file is reprocessible), merging the corrections, performing hyphenation and justification again to the extent required, and outputting to paper tape (or storing for further reference and subsequent outputting).

The various typesetting programs offered are known as "Typeset-8" systems, and can be modified, as we have said, to meet user requirements. We know of no pagination solutions, and some of the correction procedures, where they exist at all, are simple-minded. But for limited-range devices, or for special-purpose applications, where throughput speeds do not have to be high, they offer the promise of unusual efficiency. With the application of imagination and ingenuity, allowing sufficient processing time for overlaying of core so that additional functions can be performed, virtually any kind of typesetting activity could be carried for even complex work, provided it can be processed in a batch mode.

At the very cheapest level (so far as photocomposition is concerned) consider, for example, the program which Digital offers for the Compugraphic 4962. This second-generation device is a computer slave. It sells for $4950, and it will set type in a range of from 5 to 12 points, with line lengths up to 33 picas, at a rate of 35 to 40

newspaper lines per minute, from a 2-frame, 6-level computer-generated tape. (This means a throughput speed of 55 characters per second.) Two duplexed pairs of 90 character fonts may be used. The smallest ($11,845) PDP-8 system can be employed if the typographic functions to be performed are not too elaborate.

Gear changes are made manually by the Compugraphic 4962 operator in order to control lens carriage escapement. The program must be directed to use the correct character escapement values for the gear size that matches the fonts on the film strip specified. The machine monitor is notified by the use of a stop code on the paper tape and a visual message on the tape itself. A special universal gear (6-set) can also be installed in the phototypesetter and used to avoid the problem of manual adjustment.

Some of the software features include word spacing and letterspacing in order to justify text. Hyphenation can be suppressed or activated. An exception word dictionary stored in the PDP-8 is first searched before logic rules are invoked to determine when and how to hyphenate.

Column width changes may be specified. (The command $c22.3, followed by a carriage return, calls for a measure of 22 picas and three points). The $l would call for the type face on the lower magazine, and $u for upper magazine. A set of five indent commands are supplied ($k, $p, $q, $s and $z). But the user may write his own indent command by specifying, for example: $i1.3,.7, which indents copy 1 pica and 3 points from the left margin and 7 points from the right margin.

A run-around, using the command $hx,y,z can call for 3 lines of normal width (x=3), followed by 12 lines to be indented (y=12) in an amount of 5 picas and 6 points (z=5.6). Hence the command would read $h3,12,5.6 followed by carriage return.

Tabular material can be set by the use of $t, and material followed by QL (the quad left key, as on a TTS) would be set flush left. That followed by QC would be centered in the same line (if it will fit) and that followed by QR would be flushed to the right (if it will fit) or placed to the right on the next line. Leaders may also be generated between the elements by the use of $d rather than $t.

Single lines can be made to justify. The program can also ignore words which contain a "delete word" code.

Of course this program assumes input will be clean and that the program will not "hang" because of faulty codes.

### Only an Example

The use of the Compugraphic 4961 would probably provide one of the cheapest computerized typesetting systems for photocomposition, when combined (as indicated) with the lowest-priced PDP-8 typesetting configuration. With the purchase of two keyboards at perhaps $2400 each, one would be able to obtain output for less than a $40,000 investment, which could far exceed the productivity of six or more linecasting machines, with substantial additional capacity available merely for an investment in more keyboards.

More expensive photounits, with greater flexibility, and more elaborate computer systems should, of course be considered. The only point we wish to make is that the initial risk is minimal, and adding new capabilities and capacities can be modular.

A careful inspection of all types of photounits is clearly indicated and is beyond the scope of this book, since we are primarily concerned with principles and concepts.

**IBM 1130 and Related Systems**

In 1963 and 1964 the IBM 1620 was applied to the task of performing hyphenation and justification in one-pass and allotting systems, outputting to TTS-equipped readers on linecasting machines. The 1620 was not a particularly logical computer for this purpose; the "word" which comprises its 20k to 60k capacity consists of four-bits, which means that two core locations are necessary to store each character. But the 1620 had the advantage of being inexpensive, since rentals for the main frame began as low as $1200 per month.

For the most part 1620 users shifted to the IBM 1130, and this device also won many new converts. Announcement was made of this product on February 11, 1965, with a press release which stated that "it will be promoted as the basis of a relatively economical computerized typesetting system for the graphic arts industry."

> The new IBM 1130 system is general-purpose in nature, and can handle hot-type, photocomposition and other types of applications. Rental for the system, basically equipped for typesetting and other graphic arts production assignments, will begin at $1220 per month.
>
> Used with the IBM Type Composition program, which will be made available to users, the 1130 can produce fully-justified and hyphenated paper tapes for automatic typesetting at up to 12,000 30-character lines an hour. . .
>
> Another important 1130 feature is a new storage device which gives the computer direct access to any number of type fonts, as well as to an exception-word dictionary. This is said to provide the computer with a hyphenation accuracy on the order of 99 percent, according to Webster's first choice.
>
> The storage device makes use of a single magnetic disk encased in a plastic cartridge. Known as the 2315 disk cartridge, each disk is capable of storing the equivalent of more than one million characters of information. The information can be instructions for computer operation, the data to be processed, or both. . .
>
> In addition to paper tape input/output units and the 2315 disk cartridge storage, the 1130 can be used with a card read-punch and a printer. These provide the system with its ability to handle non-production applications. An 1130 equipped for typesetting applications and renting for $1,220 a month will cost $55,305.
>
> Deliveries are scheduled to begin in the fourth quarter of 1965. . . . The IBM 1130 Type Composition Program is scheduled to be available during the first quarter of 1966.

One of the early purchasers of the 1130 system was a commercial printing and typesetting firm, known as Radnor Graphic Arts, in Radnor, Pennsylvania. Because it could not get immediate delivery of the 1130 in early 1965 it agreed to start with the 1620.

V. Stuart Fuller, Jr., then Administrative Vice President of the Company,[2] reported to a PIA Computer Section meeting in the spring of 1967, in part as follows:

> Due to the unavailability of software for the 1130 system our 1130 was delayed until October, 1966. Again IBM assured us we would be pioneering on this machine. We spent three months debugging a pre-released program and we had to agree with them—we were pioneering.

Nevertheless, he said, our 1130 arrived in mid-October of 1966 and worked to our satisfaction on live jobs the day that the engineers turned it over to us. Briefly, our machine configuration now as then consisted of an eight thousand position main frame, 512,000 position magnetic disk for additional online storage, two teletypesetter six-channel readers, two typesetter six-channel punches, a card read punch and a

[2] Proceedings available in loose-leaf form to members of GCCA, of PIA.

printer... The machine is sufficiently fast so that it effectively overlaps the input of the six-channel readers with the output and thereby approximates very closely the top speed of these units of 110 characters per second.

> We feel that our computer installation has been a success. For statistics, our automatic linecasters set at 11½ lines per minute. We were averaging about 6½ lines per minute during peak periods. Also, as near as we can tell, our perforator operator speeds are up about 40% over justified copy, and these increases we experienced within the first few months of the 1620...
>
> The use of less skilled, and therefore lower cost personnel in production because of the computer has never been successful with us. However, I think that the computer's ability to increase the productivity of our skilled personnel is perhaps more meaningful....
>
> With the 1620 our total department cost was $2500 per month. It went to $5500 to $6000 for the 1130 including labor, rentals, overhead, administration, supplies, etc. We hope to start photocomposition by the end of this year.

Fuller described his hot metal operation as consisting of input from blind keyboards which are used alternately for justified or unjustified setting. Their machines do no "mixing." He contended that they had fewer corrections from the 1130 than directly from TTS input.

Another hot metal 1130 user who gave a favorable report was J. van Ginkel, of the *Het Parool,* in Amsterdam, at the 6th Technical Conference of F.I.E.J. in Paris in November, 1967. This is also largely a hot metal operation, but with mixers. He stated that "at present approximately 140,000 lines with 30,000 hyphenations are processed by the computer weekly. The number of keystrokes punched is approximately 5 million. Direct savings in the teletype room amount to 25% fewer men and, in addition, at least another 7½% in more work that we are doing with the 25% fewer men. Measured over a week, our good keyboard operators, when setting straight matter, attain a rate of 18,000 keystrokes an hour."

The IBM 1130 comes as an 8k machine with 16 bit words. Memory access time is stated as 3.6 microseconds, in comparison to the 1.6 of the PDP-8.

It is probably safe to say that many users, especially the hot metal typesetters, have not employed the 1130 for typesetting except in a single-pass concept. Very early after its hot metal programs were made available, photocomposition programs for the Linofilm, the Photon 713 and other similar second-generation machines were developed, either by IBM itself or by software houses. A correction program was developed which involved reading in the unjustified paper tape, performing hyphenation and justification and outputting the conventional paper tape. However, a printer was added to the configuration, and the printout could be proofread and entire lines substituted. These corrections would then be fed back into the system on paper tape and stored within the system so that when the output paper tape was fed back into the system the corrections were substituted for the lines that were to be replaced.

No pagination capability is offered, although certain users have developed special-purpose page makeup solutions for very standard repetitive jobs. One such company, known to this writer, is the Colwell Company, in Champaign, Illinois.

One of early users of the 1130 in the newspaper field was the Washington Star, which thereafter substituted an IBM 1800. This is a process-control type of computer which has sufficient similarities to the 1130 that it uses, basically, the same assembly language. Programs written for the 1130 will generally run on the 1800, but not necessarily vice-versa.

The Washington Star apparently found the 1800 to be a more flexible machine for its purposes, and the newspaper substantially extended the 1130 software. It then formed a company known as TalStar Computer Systems, Inc. (in July, 1969), being a joint venture between the Star and the Talcott National Corporation, a commercial finance and computer leasing firm. TalStar was claimed to be "the nation's first computer systems organization dedicated solely to the automation requirements of newspapers and related communications industries." (Press Release, October 22, 1969.) It was decided to move away from the IBM 1800 to a computer "designed and manufactured for TalStar by General Automation, Inc., of Orange, California." TalStar offers software and systems which generally run on 16k of 16 bit words (18 including parity and storage protection bits). Larger and smaller main frames are possible. A high-speed disk of the 2311 type, storing approximately 5 million characters, is generally a part of the system. The hardware claims a 960 nanosecond memory cycle time, an automatic priority interrupt system, and multiple addressing modes. Thus multi-programming is possible. At present, up to four paper tape readers and four punches have been installed in systems they have contracted for, and at least one multi-programming application permits two readers to process input from two different jobs simultaneously. Since output is usually a Burpee punch at 110 characters per second, this indicates the level of throughput speed generally provided, unless there is multi-programming.

Output programs are available for hot metal and various second generation photounits, including Compugraphic, various models of Photon equipment (including the 560—for which as many as twelve frames of information must be provided to convey width values for large point sizes) and the Linotron 505 (a paper tape driven model). At this point, although magnetic tape output would be possible, no one has yet set forth this requirement.

In addition to hyphenation and justification in a single-pass system, an alternative exists which permits the use of a printout, the storage of the input on disk in input format, and the incorporation of corrections by the substitution of one or more complete lines at the time of output. Systems offered range in monthly cost upward from $1800 per month, depending upon the configuration. The treatment of classified ads and display advertising are two important features of the TalStar system.

It is obvious that these descriptions of mini and small computer systems is sketchy. New mini computers are presently under development. New typesetting software for minis is also constantly being written. Certain companies have developed capabilities to provide software and systems and even to assist in installation and training. Taken in conjunction with the wide range of photocomposition units presently available, the alternatives available to the typesetting firm are as bewildering as ever.

CHAPTER XXIII

# Large Computer Typesetting Systems

Large typesetting systems, as of today, consist of high-speed CRT devices and medium-scale computers. The computers in most common use, as we have seen, are the IBM 360/30 and 40, or the RCA Spectra 70/35 or 45. There is a Honeywell 1200 installation, and a Honeywell 200. The Government Printing Office uses a 360/50; the Air Force operates with the assistance of some special-purpose hardware, in addition to IBM and Univac equipment. Rocappi, as of this writing, is still using its RCA 301. At least two or three Linotron 505s are being operated with smaller computers. There are several Univac models which have been devoted to typesetting, and General Electric computers have also been used.

But it is the IBM and the RCA Spectra computer typesetting applications which are presently most relevant, since the most common systems, to drive either the Harris Fototronic CRT or the RCA Videocomp, were developed for these computers. RCA offers PAGE-1 and PAGE-2 both for its own Spectra equipment and also for the 360/30 or 40. PICA (a software and typesetting systems house) has its version of the RCA Composition package for the IBM 360, and Harris offers its HCL program for the 360.

Initially PAGE-1 was written for the Spectra 35 or 45, and it ran only under its own unique operating system. The program required a minimum configuration of 65k, and it consisted of an input pass, a composition pass and an output pass. In addition, there was a correction pass which worked against the output of the composition pass.

PAGE-1, as ULTRA-X, was adapted by PICA to run on a 360 computer under DOS (the disk operating system). RCA has more recently supplied a new version, known as PAGE-2, which presently runs on Spectra equipment under TDOS (tape-disk operating system) but has been designed to run also under other types of operating systems. RCA plans to offer its own version of OS later in 1971. Although PAGE-2 was initially conceived of as a 128k program it will run in 65k but not under multi-programming.

Aside from a number of minor improvements, the two major changes between PAGE-1 and PAGE-2 have to do with the fact that PAGE-2 will run under TDOS rather than its unique operating system (and ultimately, it is claimed, could be run under other kinds of operating systems), and second, there has been a change in the method of handling corrections so that corrections are now applied against the initial input and not against the output of the composition pass. However, it is customary to obtain a printout of both the initial input and the output of the composition pass (the latter could be type galleys) and to transfer corrections back to the input printout in order to identify their location.

RCA has indicated that it is in the process of developing new software to be run on the 1600 controller which drives the Videocomp, thus dispensing with the need for a general-purpose computer for typesetting purposes. This concept has moved past the design stage but is only in early development at this point of time.

Although the Harris HCL program is not the same as the RCA PAGE-1 or PAGE-2 programs, it bears certain striking resemblances, especially in its composition language. This program was originally developed by Computype and was known as the JCL (Journeyman Composition Language). When it was sold to Harris some modifications were made, especially by way of "cleaning-up" certain routines and increasing the throughput speed. It is this program which Harris used to run its tests to gather data for the Lannon report.

The HCL program runs under DOS, and with 128k of core some multi-programming is possible, especially for the input and output passes.

### Pre-Processor Passes

Because of the complexities of these compiler composition programs it has become quite common for users to write special "pre-processor" passes which simplify input keyboarding or which format data in certain ways to interface between data processing and composition programs. These passes are usually developed by in-house programmers. Pre-processors are often the most efficient way to make the basic composition programs perform certain classes of work efficiently. But sometimes they are resorted to merely because in-house programmers do not have a sufficient understanding, and are also not generally provided with adequate documentation, of the basic composition programs. Certain standard pre-processors are available. For example, Keymatic Data Systems supplies such a package to interface between its keyboard and coding structure (which is designed to simplify input) and these composition languages.

More recently, in an effort to make its system more cost-effective, RCA has supplied a line printer program which bypasses the computer entirely and feeds directly into the Videocomp controller. In this fashion it is able to substitute type composition for line printer output on a line-for-line basis.

### The Role of IBM

Although IBM has been in the composition software business from the very beginning, its contribution to the industry's software and systems has been extremely modest to date. As we have already noted, it did develop and supply a single-pass and an allotting system program for the 1620. Subsequently it offered a program for the 1130, with a number of different output modules. But for the 360 series computer it seemed reluctant to move. Presumably there were many other challenges and demands upon its time which were either more compelling or more profitable.

By virtue of its size and the large number of its computers in common use, publishers and typesetters (and indeed other competitors) may have refrained from their own development activity in the belief that IBM would shortly provide all the software that would be necessary, and on the assumption, no doubt, that it would be more efficient because of the resources that could be brought to bear in such development.

But IBM moved very slowly. Its first contribution was merely a hyphenation module (called Hyphenation 360) which could be incorporated into a composition program. It then offered "Composition 360," which incorporated the hyphenation program,

and which performed the basic functions of hyphenation and justification, but which did not tackle input, nor provide output to any particular typesetting device, nor supply the user with any method of making corrections.

Since an Administrative Terminal System had already been developed for on-line input and corrections it may have been presumed that this would be the vehicle to handle the problem of file updating and text revision. But IBM offered no guidance with respect to ways of bridging the gap between ATS and Composition 360. In fact, the ATS system is so complex that making modifications in it to enable it to become a more appropriate composition input vehicle has become a major challenge (and perhaps a not very fruitful one). There was also some thought that the Office Products Division's MTST might supply the solution both to input and editing.

In the late fall of 1970 IBM finally furnished details of the "System/360 Text Processor PAGINATION/360 Application Description." This is Program Number 5736-K12. The explanation of its function reads:

> This program, in conjunction with user-supplied input/output programs, provides a comprehensive text processing, editing and page makeup system. It contains the features of COMPOSITION/360 (360A-DP-08X), HYPHENATION/360 (360A-DP-07X) and EDIT/360 (5736-K11), plus significant page makeup functions. Through the use of Text Processor-PAGINATION/360, the justification, subsequent modification and page composition of textual data can be performed.
>
> PAGINATION/360 will perform the operations associated with the page makeup process. These operations include the classification of input material by type (text, footnotes, graphics, etc.), the generation of page space requirements by input category, and the resolution and assignment of final page position based on user-defined parameters and page dimensions. The program can selectively reset pages whose format or content has been changed, and several different forms of justified and page output are available to meet the broad range of potential users.

While this new program constitutes a significant addition to IBM's capability, it still does not comprise a *system* since there are three signficant omissions.

1. There is no input module.
2. There is no definition or inplementation of an editing input language.
3. There is no output module.

Nevertheless, it appears that IBM has now come most of the way toward the development of a composition system.

PAGINATION/360 requires a 76k partition. But the total core requirement is 128k, since the DOS supervisor will require 10k, and it is anticipated that the balance will constitute the "user's area." In addition, there are certain utility programs:

- a font utility program of 44k
- a composition style descriptor utility program of 30k
- a file initialization utility program of 8k
- a hyphenation testing utility program of 16k for a 2311 disk or 20k for a 2314 disk.
- a dictionary maintenance and display utility program, ranging from 18k to 28k.

These utility programs are of course supplementary to, but necessary for, the utilization of the composition package.

In addition to a main frame of 128k, a number of disks, disk drives, and channels are necessary. Three 2311 disks or one 2314 would be essential in the minimum configuration along with a printer, appropriate paper tape and magnetic tape devices and whatever communications links may be desired. "The user must select these input/output devices and provide the necessary support programming."

### Two Languages Provided

Two composition languages are provided: one is known as the hot metal or "dollar sign" language. Its command structure is relatively simple and it is similar to the language used by the 1620 and 1130 programs. The second is the photocomposition, or "supershift," language, and it is the latter which must be used if pagination is desired. This presumes that pagination would not be appropriate on a hot metal device.

### A Disk-Oriented Solution

The program assumes that however input is fed into the system it will be stored on disk. Perhaps it will be dumped from magnetic tape, and may ultimately be dumped back onto tape. An important part of the overall task is the management of these files. There will be an input take file, a layout file, an output page file, an output request file, an output take file and a text storage file. In addition to the user-written input take file, there must be an input take locator file, also user-written, which indicates the location and status of takes to be processed on the input take file and of takes to be released from the output request file.

The Text Storage File is part of PAGINATION/360 and it contains, in addition to the text and command records, the text storage font sequence records and the text storage block and line locator records. These are fixed-length records with chain addresses.

The development and handling of these files is the key to the possible flexibility of the system. A number of interesting text management capabilities may be available. An input edit request makes it possible to secure a listing of the user's input takes and block numbers, line numbers and command indicators, showing the position of commands within the text, in sequential order. A command edit request can provide a listing of all of the user-entered input commands. "No original text commands are executed and no input text is justified during the command edit processing. The command edit listing is written on the Output Request File, from which the user's output module must obtain and output the command edit list."

An output edit request would permit output of justified text, once the user has provided his own output routine. A galley output request would result in the output of justified galleys. An index request would list the status of every take on the text storage file. Again the user's output module must be able to obtain and output the index list.

Editing can take place once output modules are provided. "The flexible design of the IBM/360 Text Processor permits the addition of user-written programs running in a simultaneous environment." The text processor communicates with the direct access storage device, but the user must get into and out of these storage areas. Text alterations are possible but "the user is responsible for defining and implementing an editing input language." It is presumed that this language would include "add," "erase," "replace," "move," "copy," "exchange" or "scan replace."

It may be seen from this brief description that it is difficult to describe the capabilities of this set of programs, let alone to evaluate them. Only in the context of a substantial number of user-contributed programs is it possible to understand how a system could be constructed. At any event, these IBM programs are modular, and this provides a significant element of flexibility. Their efficiency or throughput is so much a function of the kind of disks used, the size of core, whether a 360/40 or 50 is applied, how many channels are available in order to provide for I/O overlap, and the degree of intelligence applied by the user in his own system organization, that virtually no generalizations may be made. For example, these two paragraphs indicate the nature of the problem of evaluation:

> Operating the I/O modules and PAGINATION/360 in an interactive environment rather than a batch mode should result in an increase in total throughput because of overlapped processing and elimination of operator intervention. However, if the I/O module is in Foreground 1 and contains a significant amount of processing, the PAGINATION/360 partition may be locked out.
>
> The assignment of PAGINATION/360 data files to physical drives has an effect on performance. The most efficient file arrangement is one that minimizes interpartition file contention (between the I/O modules and PAGINATION/360) and intrapartition file contention (between routines within PAGINATION/360 itself). File assignment exists for all jobs. The optimum arrangement for a composition-only run and the optimal arrangement for a pagination-only run will differ because different files are used in each section.

The number of takes also affects job throughput speed, as well as the frequency of commands and PAGINATION/360 features not specifically associated with commands that are entered with the text.

Batching input and composition during one time period and page makeup during another is recommended for faster job throughput. Moreover, "PAGINATION/360 will run faster if the system contains only the documents that need to be used. The user should consider deleting documents that are not currently being used."

A list of commands for the photocomposition language is supplied. Supershift "in" and "out" codes provide delimitors. In some cases the closing delimitor is not necessary, thus saving keystrokes. For example, command< cc25,8,10> would indicate a column width of 25 picas, and would specify type face and size. On the other hand < ah would mean *allow hyphenation* until further notice, while < xh would cancel hyphenation. The command < al would allow letterspacing and < xl would cancel letterspacing.

Page makeup routines provide for a use of "keeps" or special formats. After lines have been justified and stored in the Output Take File (they may first have been edited from intermediate storage in the Text Storage File) they are made up into pages and placed in the Output Page File. Flowing text is processed by a "non-STAP" (no Specific Text Associated with Page) mode. A STAP mode (Specific Text Associated with Page) is also possible.

Layout specifications are entered via a Layout request and are converted by the Layout routine into records for the layout file. A page format pool is compiled, made up of page formats, page style descriptors and a graphic arrangement table, which may apply to several jobs. Certain page content control codes are entered into the text stream, defining copy blocks, points of leading, headings, footnotes, footnote references, and the like.

**In General**

The writer does not know anyone who is actually using the new IBM Text Processor PAGINATION/360 program, although several firms experimented with it during its developmental stages. Presumably it is being used by IBM for certain of its own publications. It is still too early to indicate what its power and capability may be. It is obvious that the program must be extended and developed by the user in relation to his own needs and equipment. This may be an asset rather than a liability, but it will not appeal to those who desire a "turnkey" operation.

Readers of this and certain of the earlier chapters may be overwhelmed by the apparent complexity involved in understanding, let alone implementing, the software and systems which have been developed for high-speed photocomposition. It should be evident that general programs which try to solve all problems must be complicated. They may be more readily grasped and applied in specific applications. As one becomes familiar with their capabilities, and becomes accustomed to working with them, what appeared impossible to grasp in the early stages may come as second nature. But, like most languages, lack of exposure and frequent use soon takes the edge off of one's competency.

One is impressed, however, by the possibility of simplifying even complicated general-purpose programs by developing higher level languages, or macros, which, in turn, will call out the more elaborate instructions or statements. It is to be expected that with greater familiarity will also come this kind of simplification.

At the moment none of these "big system" composition programs can be implemented without a substantial learning period and the development of a specialized skill which can only come with intensive application and a great deal of practical experience. One suspects that all of them are unnecessarily complex: that more decisions could have been left to the program rather than to the composition "programmer"—that is, the person who formats the input for the composition pass. If the person who outlines his preferences can do so only once, and then leave it to the computer to implement them as best it can, the task becomes simplified. It also appears that the IBM solution, working from justified lines, may be easier to cope with than the PAGE-1 or -2 compiler type of program.

CHAPTER XXIV

# Looking Back and Looking Forward

In the IBM publication which we relied upon in the preceding chapter to provide us with a summary of the System/360 Text Processor PAGINATION/360 Application Description, published in its second edition in July 1970, there will be found some extremely interesting observations, by way of introduction, on the relationship between information processing and text composition. Five points are made which are worth quoting verbatim:[1]

> The computer's function as an aid in the composition of printed text is becoming secondary to its use in processing the informational content of the same text. Printing and composition functions are increasingly considered to be implied by-products of having the text in computer-processable form.
>
> The origination of input data that eventually requires printing is not always under complete control of a specialized text processing system. The economy of being able to capture and process data from many sources is becoming more evident.
>
> The eventual form of printed text or its disposition is not always known at initial processing time. There is not only an increasing need for variations in output media and format, but also an unpredictable variance in the elapsed time period between processing activities on either the text or the information it represents.
>
> New composition devices and computer system devices that enhance the role of the computer in a text processing environment are becoming available as advances in technology are combined with a better understanding of the problem by computer users and manufacturers.
>
> Computer applications relating to text processing are multiplying, thereby increasing the requirement for additional computing power to handle them. For economic reasons, this trend has dictated a requirement to use the text processing computer system for nonrelated data processing functions, especially during periods of low text processing activity.

We shall have some trouble, then, reconciling the probable trend toward the creation and use of input for informational purposes as differentiated from (or in addition to) text composition purposes. This is because the informational aspects of the challenge seem to require large computer system solutions, with their attendant complexity and the problems of the economics of scale, and the composition aspects can best be performed on small systems, not only for reasons of economy but also because the systems are less complex and more readily implemented by dedicated persons of normal intelligence.

Over the past seven or eight years there have been parallel developments of large and small systems, although most of the attention has been focussed upon the larger

[1] Program Product GE20-0328-0, p. 1.

ones. These installations, as is now well-known, encountered difficult times. Harry Sedgwick spoke very bluntly at the first event of any industry significance which took the plight of the industry into account. This was the Research and Engineering Council meeting, the so-called "International Conference on Computerized Typesetting" in June of 1970.

> To say that our industry is in financial difficulty I think is an understatement at best. . . . I wouldn't suggest that the whole activity is going down the drain, but I would suggest in all seriousness that computer-controlled typesetting is in bad trouble unless we wake up.
>
> The tone of this 3rd International Conference, to date, has been somewhat different than the 2nd Conference in October 1967, when we all got together and congratulated ourselves about how much money we were going to make and how we were going to have service centers all over the country and pages cluttering up the countryside, and that we'd laugh our way with the truckloads of money to the bank.

Sedgwick described some of the factors which helped to create this plight.

> First, we allowed the trade journals to do our market planning for us. For ten years you couldn't pick up a trade journal without reading that you really ought to hustle into computer typesetting because you're going to turn the world around. And we believed it.
>
> That chorus was joined in by the computer manufacturers who, needless to say, even if they had known anything of the nature of the market, weren't about to tell the market that the services from these devices were extremely limited and questionable in economics at best. . . .
>
> Secondly, I don't think we knew the size of the market; that is, the size of the ready market. If you do a job of analyzing this market for computer typesetting on the basis of SIC codes and Harvard Business School tactics, forget it; although there was something like $500 million spent in typesetting last year coming out of 1,600 trade shops, that's $360,000 per trade shop, by the way. . . .
>
> We all know now that if software doesn't work it (the system) doesn't operate at *any* characters per second, that if the equipment is down it doesn't operate at *any* characters per second. But what is much more important, if the business isn't there, it operates at zero dollars a second. . . .
>
> Here we are. We are in deep financial distress. There are a number of our colleagues in this business who are looking for financing today, and in this market they aren't going to find it. It's even difficult to find financing for a flourishing enterprise. It is extremely difficult, if not a fatal time, to go out for money, particularly when you're at your second or third financing. And the promises that management gives to directors and directors give to either actual or potential stockholders wears somewhat thin after a four-year period, or whatever the period has been.[2]

At the time of Sedgwick's speech, and for some months thereafter, a shakedown began to take place in the industry, but the sales of small photounits and small systems continued.

There can be no doubt, therefore, about the future of these small systems. And while they may provide a new base for the creation of sophisticated approaches which will truly address themselves to the informational (or non-compositional) values which have been indicated, there is still need for the development of a larger systems concept.

[2] Proceedings of International Conference on Computerized Typesetting, June, 1970. Henry D. Sedgwick, p. 57ff.

As the writer suggested in his paper to the Composition Workshop (see Chapter XIX, page 186 of this book) the problem is one of systems and software. While input hardware needs improvement, and terminals for editing need to be refined, we now have many splendid output devices and will shortly have others at lower prices, with equally great flexibility. There will be an output device to meet virtually every need.

However, we must continue to address ourselves to systems and software problems. Today there is no software package that is not proprietary, with the possible exception of that used by the Government Printing Office. Moreover, these packages have been written for special configurations of equipment, usually in nonportable computer languages. Where they have been written in a compiler language they are inefficient.

There is very little universality in concepts, codes, commands, or hardware requirements.

Very little new work is being done in the software field at present, except for minicomputers. Manufacturers do not seem to be committed, presently, to any major new software systems, nor do users.

If a common assembly language is used, such as IBM/360, compatibility may be assumed in view of the trend toward common instruction sets, emulators and simulators, programmable instruction stores, and macro-programming techniques. However, certain routines could be written in a compiler language where efficiency is not sacrificed.

The operating environment must assume multi-programming, multi-job processing and at least limited time-sharing. Hence programs should be re-entrant.

Programs should be written as small modules which can be linked together to form the complex desired. The difference between general-purpose and special purpose programs will be determined largely by the manner in which these modules are joined. Hence the appropriate degree of efficiency for each major job can be pre-determined.

Programs should be written to be independent of any particular hardware, either for input or output, and it should be easy to move from one operating system to another, taking as much advantage as possible of the features of particular operating systems.

There should be no presumption concerning code structures, except for the establishment of conventions for grouping symbol sets in logical structures.

The system must provide for interaction between the user, the program and the computer. Reprocessability at all stages and cycles is essential. It should not be necessary to revert to input.

Redefinition of all elements in any job must be possible at any time, and the content of the codes should be external to the text stream, being available on disk for reference, for correction, and for revision.

Edit programs must be interactive, reprocessable and highly flexible. Error checking, verification of instructions and format conditions should be a feature of all programs, but especially the edit routines.

Interaction may be less desirable for page makeup, but for book work conditional statements should be avoided since the program should have full discretion, with the capability on the part of the user to override its decisions.

All typesetting programs should contemplate the ability to accept data processing files or to create data processing files from typesetting input. Such files should be

processable on the basis of manipulation of information *about* the data rather than of the data itself.

The writer tends to believe that truly good software does not exist, and that failures in systems concepts are responsible for most of our industry's economic problems today. "Enough time has gone by, enough water has swept over the dam, that we now know enough to design a good system. Equipment is good enough and reliable enough, and economical enough, that if we can bring the right software solutions to bear we should reap great benefits as persons concerned first and foremost with the widest possible dissemination of information, in the belief that knowledge and understanding will promote progress, peace and justice. . . . Unless we take an ambitious stance there will be a proliferation of inadequate solutions; information interchange will not take place in a meaningful way; the results of information processing will not be readily conveyed to the possible users, and we shall have lost many of the potential benefits of present-day technology." [3]

### Relationship to Graphics

We have not discussed the implications of graphic digitization in this book, nor have we dealt with the implications of COM (computer output microfilm), or the relationship between the book and the microfiche. We have not considered the library of the future nor have we considered ways in which knowledge will be organized, input and retrieved.

These are large subjects, deserving of further exploration at another time. But it is evident that we are on the threshhold of further technological advances that are extensions of digitized character storage and character generation, on the one hand, or information processing with the aid of communicating TV-type terminals on the other.

Graphic digitization, for example, requires the availability of mass storage, whether on tape or disk, or some new digitized photographic carrier read by laser-beam technology. The only point we need to make now is that large systems and computer utility services seem appropriately a part of the developing trend, and perhaps this adds new cogency to the argument that we should not forsake ambitious, major solutions in the pursuit only of the small systems that can be more readily grasped, manipulated, and exploited economically by the small, craft-oriented typesetter.

[3] Computer Workshop, op. cit.

# BIBLIOGRAPHY

It is not necessary to provide a detailed bibliography of the literature in this field since an excellent one already exists. This is the *Bibliography on Electronic Composition,* prepared for the Joint Committee on Printing, and published by the U.S. Government Printing Office in 1970. While no date is specifically provided, it appears that material is included at least through mid-1969.

Consequently, we shall add only those works of particular significance which have come to our attention since that time.

### A. Conferences and Proceedings


1. Graphic Communications Computer Association of Printing Industries of America, Inc.

   *Selected Computer Section Conference Presentations on Character Generation* -(loose-leaf notebook available to members).

   1969 Conference: Case Histories
   - V. Stuart Fuller, Jr., Radnor Graphic Arts
   - Francis C. Corley, Corley Printing Company
   - John H. Attas, Jr., John H. Attas Company
   - Frederick L. Faulkner, Graphic Arts Data Service
   - Frank J. Weis, New York Telephone Company

   1968 Conference: The Customer Looks at Computerized Composition
   - Hilmer Stark, Billboard Publications, Inc.
   - Donald S. Sivyer, Bramson Publishing Co.
   - Frank B. Leonard, Bell Telephone Laboratories
   - Frederick L. Faulkner, Graphic Arts Data Service

   1969 Conference: The Publisher's View
   - Charles J. Kerins, TV Guide Magazine
   - Orland J. Engle, R. R. Donnelley & Sons
   - John B. Salzman, LEASCO Systems
   - Burt Hofferth, Encyclopaedia Britannica

2. 1970 Conference, St. Paul, May 13-15 (unpublished) but see especially unpublished speech of Margaret T. Fischer, Time, Inc. on conference title "The New Dimensions of Terminal Oriented Computer Systems," and newsletter summary of remarks of John W. Seybold on Lannon's report.

3. Composition Input Proceedings, Miami Beach, November 4-7, 1970, including presentations by:

John W. Seybold (consultant)
Walter Jessel, IBM
Jack Turchon, Keymatic Data Systems
Ronald Bush, CompuScan
D. D. Dissly, Dissly Systems
Dr. Edwin Kolb, Harris Intertype (Graphic Digitization)
H. A. Ulijohn, Interface Mechanisms
Harold Kneller, Connecticut Technical Corp.
Thomas J. Coad, Automix Keyboards
Donald Sweeney, Star Parts
G. A. Boucher, Fairchild Graphics
Robert Barry, Visual Graphics
Peter Kersten, Photon, Inc.
Charles Newman, Information Control Systems
Joan VanHorn, VIP Systems
Paul McGarrell, Harris Intertype (Harris Terminal)
Charles Wulfhorst, Hendrix Electronics (Terminal)
F. C. Holland, The Monotype Corp., Ltd.
Collier Elliott, Mid-Atlantic Comptronics
Frank Cremonesi, International Typographical Union
James J. Barrett, Malt Keyboard Dynamics
Kelly Hicks, Kingsport Press
Jerry C. Weissburg, R. R. Donnelley
P. Lennart Andersson (consultant)
Jerry Carlson, Farm Journal
Jean Pierre Cuny, Didot-Etude, Paris
(Proceedings published by GCAA, PIA, February, 1971.)

4. American Society for Information Sciences (ASIS) *Composition Workshop,* Washington, D.C., December, 1970. Published, February, 1971. Robert Landau, Editor. A series of papers and informal discussions.

5. Research and Engineering Council of the Graphic Arts Industry, Inc., Arlington, Va. *Proceedings of International Conference on Computerized Typesetting,* June 22, 23, 1970.

6. *IEEE Transactions on Engineering Writing and Speech,* August, 1969. Volume EWS-12, Number 2. New York, N.Y.

7. Battelle Memorial Institute, Columbus Laboratories, *COMPRINT 90,* (part of material distributed at COMPRINT 90, Sponsored by Printing Industries of America, Inc., and MGD Graphic Systems, North American Rockwell, October 12-14, 1970.

### B. Articles and Reports

Andersson, P. L., "Phototypesetting Today," *Printing Production,* December, 1969.

Andersson, P. L., "Photosetting—A Quiet Revolution," Datamation, December 1, 1970.

Carlson, Jerry A., Technical Report, "Magazine Writing and Editing on Computer Video Terminals," Farm Journal, 230 West Washington Square, Philadelphia, Pa. 19105 (*unpublished report,* 1970).

Duncan, C. J., Application of Computers to Typesetting, Report to the Printing and Publishing Industry, Computer Typesetting Research Project, University of Newcastle upon Tyne, Final Report for Period September 1965–February 1970 *(preliminary draft),* February, 1970.

Electronic Composition Cost Comparison, Joint Committee on Printing, October, 1970, U.S. Government Printing Office *(The Lannon Report).*

Lessing, Lawrence, "The Printed Word Goes Electronic," *Fortune,* September, 1969.

Hines, Theodore C., Jessica L. Harris and Martin Colverd, "Experiments with Computer-Assistant Indexing: American Documentation, Volume 20." *American Documentation,* November–December, 1970, p. 402.

Strauss, Victor (all articles in *Publishers' Weekly ):*

Lannon's Fluxions for a Technology in Flux, Oct. 12, 1970.
A Look at Two Low-Cost Photocomp Systems, July 6, 1970.
Production Editing at McGraw-Hill, October 6, 1969.
Betwixt Cup and Lip, January 26, 1970.
R & E Council 1970 Probes the Maze of Computer Semantics, June 8, 1970.
Computer-Based Composition and Editing: An Exchange, July 7, 1969.

### C. Continuing Publications and Reports

See especially the publications of *Composition Information Services, Inc.,* especially the annual surveys of installations, the CIS Newsletter, Software Central, Installations and Applications, and other releases. 6331 Hollywood Boulevard, Los Angeles, California.

### D. Miscellaneous

The author has relied upon publications of the manufacturers, especially as cited throughout.

abridging 162
abstracting 162
abstracts 197
accents, floating 17, 168
Administrative Terminal System 181
advance sprocket paper tape 23, 25
aesthetics of composition 2,11
air line schedules 197
ALGOL 80
alignment (base) 10, 36
alignment of numerals 9
allotter function 86
alpha characters 9
alphabet 141
Alphanumeric 51
alphanumeric data 65
American University 201
analysts, systems 75
area composition 100
arithmetic and logic areas 64
ascenders and descenders 13, 14
ASCII 70ff
ASIS 186
ASR 33 Teletype 211
assembly elevator 21, 31
assembly language 80, 225
Astrocomp 211
ATF (American Type Founders) 28, 32, 41, 129
ATS (Administrative Terminal System 181, 219
author-publisher relationship 59, 60, 188
author, relationship to typesetting 56, 60
author's alterations 56, 59, 193
Auto-Logic 52
automatic indexing 162
automatic quadder 10
back-up, importance of 124
bank men 56
Barnett, Michael 133
base alignment 13, 17, 35
BASIC 80
batch processing 136
beam deflection 49
bell or stop code 23
Bell System 153
Bezanson, Howard 115
bibliographic citations 197
binary code 23, 65ff
bit configurations 24, 25, 65, 66
blind keyboards 22, 87
book catalogs 197
boundaries 100
"branch" 78
"burst" mode 64
by-products of typesetting 151ff
bytes 65
carding 132, 135
cards, punched 63, 154
Carlson, Jerry 57, 187
casting 4
cathode ray tube (see CRT)
central processing unit (cpu) 63
channels 21
channels (linecasting magazines) 32
channels (computer) 63
chapter openings 135, 174
character (definition of) 141
character density (see character resolution)
character digitization 45, 47
character duplexing 27
character enlargement 17
character generation 41, 47
character positioning 16
character repertoire 8, 22, 24, 25, 29, 31, 36, 37, 49, 51, 65, 87, 140, 165ff, 212
character resolution 44, 47, 49, 50
character scanning 47
character sets 8, 140 (see also character repertoire, extended character sets)
character storage 43, 45, 47, 49, 51, 65
characters, alpha 9
characters, non-alpha 9
characters, width of 1, 4, 22, 34, 85
"cleaning up" characters 47
COBOL 80
code compatibility 139, 225
coding 60, 66, 80, 127, 165ff, 174, 225
cold composition 1
cold type 7
Columbia Broadcasting Laboratories 43, 54
Colwell Company 215
column-merge 130
COM devices 225
communications industry 177
compatability of codes 139
compiler languages 80, 225
compiler programs 134, 153, 222
complexity of typesetting 55
composition 100
composition, area 100
composition, cold 1
composition, definition of 1, 100
composition functions performed by photounits 41
composition, impact 1
composition, hand 2
Composition Information Services (CIS Newsletter and other publications of Technical Information, Inc) 134, 195, 196, 201
composition, machine 2
Composition Systems, Inc. 211
composition, typewriter 1

Composition Workshop 186, 225
CompuComp 134
Compugraphic 210
"compute bound" 84
computer concepts 63ff
computer output microfilm (COM) 225
computer processing unit (cpu) 64
computer slave 31, 41, 212
computer software 64, 75ff, 83, 211, 225
computer typesetting 83
computer typesetting, economic aspects of 189ff, 209
computer typesetting installations 201ff
computer typesetting systems 83
computer typesetting, transition to 26
Computype 218
control unit 64
controllers (computer) 64
core 65, 81, 99, 212
core partitions 92
core requirements 116
core "swapping" 92
conditional statements 133
conventions and standards 140, 143
corrections 58, 60, 85, 87, 88 89, 116, 119ff, 181, 194, 215, 217
corrections, cost of 59, 60, 192
corrections, problems of photo-composition 30
copy cutting 86
copy editing 57, 177ff
copy markup 18
cost of corrections 59, 60, 192
counting keyboards 4, 36
counting magazines 23
CRT 44, 50, 151
CRT terminals 57, 100, 123, 136, 184
CRT tube costs 50
CRT tube distortion 50
CRT tubes 44
CRT typesetters 16, 19, 47, 151, 193, 209, 217 (see also names of equipment)
cutting copy 29
data manipulation 73, 225
data processing 151ff, 223, 225
Datatext 182
data transmission 63
"de-bugging" 80
decimal numbers 67
DECtape 211ff
"deficit" in line calculations 27, 98
delete codes 22
delimitors 107, 221
density of characters (see also character resolution) 50
descenders 13, 14
device-independent programs 225
diagnostics 81, 85, 120
dictionaries 180
dictionary look-up 116, 179, 211
Didot-Berthold system of type measurement 18
Digital Equipment Co. 204, 209ff
digitization of fonts 51
digitized characters 45, 47, 49
digitizing graphic images 50
diphthongs 167
disks (type) 32, 33
disks (computer storage) 45 47, 51, 65, 89, 100, 135, 212
display devices 122
display heads 95, 221
distortion 45, 54
distributor 21
documentation 81
Dolby and Resnikoff 180
Donnelley, R. R. 136, 184
drum storage (film) 34, 35
drum storage (computer) 65, 89
duplexed fonts 34
duplexed mats 32
dynamic leading 51
EBCDIC 70ff
economics of computer type-setting 189ff, 209, 223
editing 119, 126, 136, 177ff, 212, 220, 225
editing for format 184
eight-level tape 33
electron beam 44, 49
electronic composition cost comparison 193
electrons 54
elevate 23
Elliott Automation 83
em, en, thin 9, 23, 87
em space 9, 15, 18, 95, 96
emphasis, typographic 7
emulators 225
Encyclopaedia Britannica 180
end-of-line decisions 4, 22, 26
end products 55
equipment 11
escape codes 145
escapement 1, 16, 31, 32
Evans, Harold 210
exception word dictionary 116, 211
explosion 162
extended character set 51, 165ff
extract 29
extraction 162
facing pages 131
Farm Journal 57, 187
F.I.E.J. 215
fielded data 154
file creation 151ff
film advance 19, 51
film strips 35
file handling 220
film transport shift 49
firmware 64
fixed spaces 9, 23
"flags" 95, 161, 165ff
flash tubes 43

flexibility of photounits 38
Flexowriter 35
floating white space 133
flow chart 75ff
flush (left, right, left and right) 10, 11, 103
flying dot (spot) scanner 47
folios 56, 91, 130
font designators 165ff
font digitization 51
font storage 45
fonts 166
fonts, predigitized 47
fonts, definition of 141
font sections 141
font storage 49
footnotes 29, 56, 174, 221
format 9, 91, 174, 221
format statements 148
FORTRAN 80
Fotomatic 32
Fotosetter 17, 32
Fototronic CRT 47, 51, 194, 195, 217
Fototronic Filmsetter 37
fractions 169
Friden Justowriter 28
Fuller, V. Stuart, Jr., 214
galleys 56, 91, 123, 129
General Automation 216
global notations 147ff, 174
Government Printing Office (U.S.) 225
government typesetting 50, 217
Graphic Communications Computer Association (see also PIA) 133, 177, 214
graphic (definition of) 141
graphic escape code (definition) 141
graphic images (digitizing) 50, 225
graphics 50
Greek and math setting 169
grid shifting 50
grid storage 43, 50
grid turret 36
hand assembly of type 29
hangers 58
hard copy 36, 87
Harris HCL 134, 217
Harris Intertype 31 (see also Fotomatic, Fotosetter, Fototronic, Intertype, TxT, Terminals, Digitizer)
Harris Intertype Fotosetter 32
Harris Intertype Fototronic Filmsetter 37
Harris Intertype Fototronic CRT 47, 194
Harris Intertype TxT 210
heads, display 95, 135, 174 221
height of type 13
Hell, Dr.-Ing. Rudolf 43
Het Parool 215
hexadecimal codes 66, 69
high-level language 80
Hollerith cards 154
homographs 115
Honeywell 217
horizontal rules 103
hot metal computer systems 201ff
hot metal linecasting 55, 56, 85
hot metal systems 8, 214, 215
hot type 1, 7, 56, 85
hot type design influence 2
hyphenation 85, 88, 98, 109ff, 210, 218
hyphenless justification 109, 210
IBM/Alphanumeric 2680 51
IBM "Composition/360" 218
IBM Federal Systems Division 136, 184
IBM "Hyphenation/360" 218
IBM MTST 126, 219
IBM "Pagination/360" 219
IBM Selectric Typewriters 1, 3, 4
IBM Systems Development Division 141
IBM 1130 204
IBM 1620 214, 218
IBM/360 computers 65, 194, 209, 217ff
IBM 1800 215
illustrations 141
image enlargement or reduction 45
image latency 54
impact typesetting 28
indents 104
Index Medicus 49
indexing, automatic 162
industrial directories 197
inferiors and superiors 49, 171
information processing 223
Information Control Systems 211
input 60, 85, 139ff, 143, 221
input-bound 64, 84
input modules 92
input structure 139
Institute on Technological Change 209
instructions, programmed 64, 72, 225
intaglio images 21
interactive systems 60, 136, 225
International Conference on Computerized Typesetting 224
Intertype 3, 47
Intertype Fotosetter 32
Intertype (see also Harris Intertype)
Intertype Fotosetter 32
interword spacing 3, 97, 109ff
I/O overlap 221
italic 31
Jessel, Walter 141
JusTape and JusTape Jr. 204, 209
justification 2, 3, 4, 27, 28, 29 31, 85, 88, 95ff, 210
justification, hyphenless 35, 210

justification, multi-line 117
justification, quality of 35, 109ff
justified setting 2
justifying spaces 3, 97
Justowriter 28, 32
kerning 16, 36
key words 181
keyboard (blind) 22
keyboard (counting) 4, 36
keyboard features 35
keyboard, justifying 22, 85
keyboard layout 8, 139ff, 165ff
keyboard longhand 145
keyboard, linecasting 21
keyboard, Linofilm 36
keyboard, separate 38
keyboard shorthand 147ff
keyboarding 22, 85, 102
keyboarding economics 190, 215
keyboarding speed 190
keyboards (typewriter) 7, 8
Keymatic Data Systems 218
knowledge industry 177
language, computer 80
Lannon, E. R., 193, 207, 209
latency of image 45
layout specifications (see also parameters) 221
leadering 11, 103
leading 14, 17, 18, 49
leading (reverse) 49, 130
lens system 36, 37, 45
lens, traveling 43
lens turret 33
letterspacing 2
library catalogs 153, 197
ligatures 9, 23
light sources 34
Linasec 204, 209
line printer, substitute for 152
linecasting machines 21, 22, 59, 215
"lineinf" 130
line justification 3, 95
Linofilm 33, 36, 96, 211
Linotron 505 45, 50, 194, 211
Linotron 1010 43, 50, 100, 130
Linotype 3, 57, 59
load tapes 35
locking codes 145, 168
lock-up 29
logic areas 64
"loops" 78
lower rail 23
machine language 80, 139, 147
machine-readable input 9, 127
machine speeds 31
macros 147ff
macro-programming 225
magazine publishing 57
magnetic tape 63
make-up man 56
main frame 63
management 189
manufacturing schedule 59
manuscript 56, 59
markets 152, 174, 197
mark-up 57
mark-up costs 170, 191
mass storage 225
master type images 37, 50
masters, photographic 50
mat case (Monotype) 27
mathematical equations 36
matrices 3, 15, 21, 59
matrix widths 32
McGraw-Hill 59, 185
measure, definition of 3
measure, length of 21, 95, 104
memory dump 80
memory store 41
Mergenthaler 43, 54 (see also Linotron 505, Linotron 1010, VIP, Linofilm)
Mergenthaler Linofilm 33, 36
Mergenthaler Linotype 21
Mergenthaler, Ottmar 21
Mergenthaler VIP 210
"mini" computers 127, 196, 210, 225
mixing of fonts 215 (see also character repertoire)
mixing of sizes 35
modularity 91, 92, 139, 213, 215
Monophoto 17, 32, 201
Monotype 4, 15, 27, 201
mortising 15
MTST 126
multi-column material 129
multi-face perforator 23
multi-job processing 225
multi-level equations 36
multi-pass systems 83, 89
multiplexor channels 63
multi-processing 80
multi-programming 80, 84, 92, 216, 217, 225
"object" program 80
obliquing 44, 47
OCR (optical character recognition) 9, 141, 144
octal codes 66, 96
"OEM" 210
off-line typesetters 52
offset printing, effect on photocomposition 29
one-pass systems 83, 84
on-line printer 87
on-line systems 60
on-line typesetters 52
operating systems 80, 92, 217
operations, computer 72
orthographic equivalents (definition) 141
output 60, 85
output-bound 64, 84
output modules 91
overlaying characters 51, 100

National Library of Medicine 43
National Science Foundation 186
Newcastle-upon-Tyne 83, 130, 207
newspapers 86, 101, 201
"no-break" areas 133, 174
non-locking codes 145
non-structured data 151ff
numerals, alignment of 9
nut space 9
PAGE-1, PAGE-2 100, 133, 217
page balancing 56, 131
page formatting 49, 174
page makeup 29, 41, 56, 59, 100, 119, 123, 129ff, 174
page makeup costs 192
pagination 41, 90, 119, 129ff, 174, 215
paper tape input 22, 32, 41, 63
parameters 95, 135, 221
parity 66
partitions of core 92
parts lists 197
PDP-8 209ff
peripherals (computer) 63
perforator, multi-face 23
Perry Publications 116
photo-multiplier 43
photocomposition 55, 85
photocomposition, advent of 28
photocomposition, computer systems 201ff
photocomposition devices, first generation 31ff
photocomposition devices, second generation 33ff
photocomposition devices, third generation 41ff
photocomposition speeds 29
photographic enlargement of type 16
photographic grids 43
photographic lens systems 16
photographic masters 50
photographic reduction 17
phototypesetters, type design for 16
phototypesetting, see photocomposition
Photon 200 33
Photon 513, 540, 560 34, 211
Photon 713 34, 211
Photon 900 and 901 41, 43
Photon 7000 52
Photon "Zip" 41, 43, 129
pi mats 26
pi characters 170
PIA (Printing Industries of America) 133, 174, 193, 214
PICA (Printing Industry Computer Associates) 134, 217
pica, definition of 18
picas 14, 96
PL-1 80
point sizes of type, range of 31
points and picas 18
point system of type measurement 14
precedence codes 23
pre-digitized fonts 47
prefix analysis 114
pre-processors 218
printer, on-line 63, 87, 121
print-out device 65, 87, 121
printouts 88
prism, traveling 43, 45
product guides 197
production techniques 57
programmed instructions 64
programming customer's data 152
programming (interactive) 136, 186
programs (computer) 75ff
proportional type 1, 10, 129
proofing mode 50
proofreading 58, 121, 149
proofreading costs 191
proofs 56, 59, 60, 121, 122, 212
Publishers' Weekly 185
publishing defined 177ff
punch bound 212
punched cards 63, 154
punched paper tape 22, 63
Purdy-McIntosh 43
quadded lines on photocomposition devices 29
quadding 10, 11, 23, 87, 96 101
quality of type 47
queries 56
queuing 60
quotation marks 9
Radnor Graphic Arts 214
ragged setting 2, 102, 105, 109
rail 21, 23
rail, upper and lower 24, 25
Random House Dictionary 180
raster 42
RCA 196, 197
RCA PAGE-1 or PAGE-2 100, 133
RCA Spectra/70 computers 194, 209, 217
RCA Videocomp 45, 217
"read only" features 64
real-time 136
readability and proportionality 1
redundancy elimination 162
re-entrant programs 136, 186, 225
registers 64, 173
reperforator units 87
repertoire of characters 8, 22, 24, 25, 31, 33, 49, 65, 87
"repro" (reproduction) proofs 29
reprocessability 125, 224
Research and Engineering Council 224
resolution of characters 44, 47, 49, 50
reverse leading 49, 130
Roberts, Irv 134

Rocappi 217
rub-outs 22, 23
rules, horizontal 103
running heads 56, 91, 130
running text 10
scanner, flying dot 47
scanning 144
schedule of manufacturing 60
second-generation photocomposition devices 33
Sedgwick, Henry B. 224
selector channels 63
set sizes and concepts 19, 99
set-width tables 99
Seybold, Jonathan 193
sequencing 153
Shatzkin, Leonard 58, 185
shift codes 23
side bearings 14
sign (definition) 141
simulators 225
six-level tape 33
slow functions (Linotron 505) 44
slug casting 3, 21
small caps 23, 24, 168
small systems 209ff
software 64, 75ff, 83, 211, 225
sorting and sequencing 151ff
spacebands 3, 10, 21, 32, 98
spacebar 3
spaces, fixed 9, 37
spacing, interword 3, 97
spacing out (vertically) 29, 130
spaces, justifying 3, 97
spacing, variable 3, 97
special-purpose applications 197
speed in composition 29, 50
standard products 198
standards and conventions 140
STAP 221
"step effect" 47
stock market transactions 197
storage areas 64
storage for corrections 89, 211, 212
storage of width values 34, 211
storage-protect functions 64
storage unit 64
Strauss, Victor 185
structured data 127, 151ff
"subject" programs 80
sub-routines 79, 83
subscripts and superscripts 49, 171
"Syntran" 181
system 11
system flexibility 51
system, operating 92, 217
systems analysts 75
systems, computer typesetting 83ff, 209ff, 217ff, 225
systems, on-line 60
table look-up 68, 96
"tags" 82, 95, 161, 165ff
Talcott National Corporation 216
TalStar Computer Systems 216
tape compatability 36
tape, incompatability of 28
tape, paper 63
tape, magnetic 63, 212
tape, storage 65, 89
tariff listings 197
telephone directories 197
terminals 57, 100, 123, 136, 181 186, 187
text, definition of 141
text editing 119, 126, 136
text (hot metal) point sizes 22
text (running) 10
thin space 9
third generation typesetting devices 41ff
thirty-one channel tape 38
throughput speeds 116
Time magazine 95, 184
time sharing 92, 197
trailer 160
translate by table 97, 159
transmission of data 63
traveling lens 43, 45
TTS, advantages of 26
TTS codes 24, 25
TTS keyboards 22, 23
TTS linecasting 22, 215
tube aperture 49
tube distortion 45, 54
Tunnicliffe, William 133
turn-over lines 102
turn-page feature 50
type character expansion 50
type, characteristics of 13
type characters, proportionality of 1, 129
type design 3, 13, 51
type disks 32, 33
type expansion or condensing 19, 50
type faces 14
type, hot 1
type, measurement of 14
type quality 47
type, range of point sizes 31, 212
type sizes (hot metal) 22
type, unit of measurement of 15
typesetting devices, second generation 33, 209
typesetting devices, third generation 41ff, 217ff
typesetting systems 55, 209
typesetting uniformity 58
typewriter, standard 2
typewriter, proportional 2
typographic emphasis 7, 152
typographic flexibility 8
typographic requirements 11
typographic situations 95, 127, 165ff, 173

typographic symbols 127
typographic variety 55
"typos" 56
Ultra-X 134, 217
uniform widths 1
unit system 4, 15, 95
updating, see also corrections 211
upper rail 21, 23
USASCAII 70 ff
vanGinkel, J. 215
variable spacing 3
variable-width characters 10, 37
verification 59, 120
verification of hyphenation 117
vertical justification 132
video terminals 57
Videocomp 43, 45, 194, 195
vowel group analysis 111
Washington Star 215
Waverly Press 115
work flow 55
"widows" 56, 58, 59, 131
width information 33, 85
width values 22, 27, 34, 85, 97, 99
widths, variable 37
words (computer) 65
word definition 110
word division, see hyphenation
word structure analysis 114
work areas 64
Wright-Patterson Air Force Base 50
"x" height of type 13
"z" height of type 13
Zip (Photon) 41